

Документация Calculate Linux

Наша документация доступна на следующих языках:

[English](#) | Russian

1. Установка Calculate

1. [Об установке Calculate Linux](#)
2. [Краткое руководство по установке](#)
3. [Установка на жёсткий диск](#)
4. [Установка на Flash](#)
5. [Аппаратные требования](#)
6. [Структура FTP-зеркала](#)
7. [Разбиение диска](#)
8. [Программное обеспечение](#)
9. [Установка с загрузочного CD-диска](#)
10. [Чем заняться дальше?](#)

2. Работа с Calculate

1. [ЧаBO \(FAQ\)](#)
 2. [Руководство пользователя](#)
 3. [Создание учётных записей](#)
 4. [Шифрование домашних директорий](#)
 5. [Установка и удаление программ](#)
 6. [Руководство по обновлению системы](#)
 7. [Сценарии инициализации](#)
 8. [Сборка ядра со своей конфигурацией](#)
 9. [Интерактивная сборка системы](#)
 10. [Системные утилиты](#)
 11. [Оптимизация системы](#)
 12. [Загрузка модулей ядра](#)
 13. [Ревизии системы](#)
14. Пакеты оверлея Calculate: [calcboot](#), [calckernel](#), [calculate-sources](#), [keyexec](#), [pam_keystore](#)

3. Работа с Portage

1. [Введение в Portage](#)
2. [USE-флаги](#)
3. [Возможности Portage](#)
4. [Переменные среды](#)
5. [Файлы и каталоги](#)
6. [Настройка с помощью переменных](#)
7. [Смещение ветвей программного обеспечения](#)
8. [Дополнительные средства Portage](#)
9. [Отступление от официального дерева](#)
10. [Использование ebuild](#)

4. Утилиты Calculate

1. [Графический клиент утилит Calculate](#)
2. [Консольный клиент утилит Calculate](#)
3. [Сервер утилит Calculate](#)
4. [Шаблоны утилит Calculate](#)
5. [Переменные шаблонов](#)
6. [Хранение настроек профиля пользователя](#)
7. [Обновление системы cl-update](#)
8. [Смена профиля системы cl-update-profile](#)
9. [Сборка системы](#)
10. [Calculate API](#)

5. Настройка сервера

1. [Перенос учётных записей пользователей в Calculate Directory Server](#)
2. [Настройка LDAP сервера](#)
3. [Использование LDAP сервера для хранения учетных записей](#)
4. [Настройка Samba сервера](#)

5. [Настройка прав доступа ACL](#)
6. [Настройка FTP сервера](#)
7. [Настройка Jabber сервера](#)
8. [Настройка почтового сервера](#)
9. [Настройка Proxy сервера](#)
10. [Настройка DNS сервера](#)
11. [Настройка DHCP сервера](#)
12. [Настройка PXE](#)
13. [Настройка репликации Samba серверов](#)
14. [Настройка репликации почтовых серверов](#)
15. [Настройка сервера шлюза](#)
16. [Настройка Asterisk сервера](#)
17. [Обзор структуры LDAP сервера](#)
18. [Управление клиентскими машинами](#)
19. [Система виртуализации QEMU](#)
20. [Настройка gitolite](#)
21. [Резервное копирование](#)

6. [**Настройка рабочей станции**](#)

1. [Переход на Linux](#)
2. [Подключение к серверу каталогов](#)
3. [Хранение пользовательских настроек](#)
4. [Установка шаблонов пользовательского окружения](#)
5. [Реализация кэширования NSS для доменной машины](#)

7. [**Настройка сети в Calculate**](#)

1. [Настройка сети Gentoo-way](#)
2. [Настройка сети Calculate-way](#)

8. [**Настройка оборудования**](#)

1. [Настройка звука](#)
2. [Настройка softmodem](#)
3. [Настройка TV тюнера AVerMedia AVerTV 305/307](#)
4. [Настройка сканера Epson Perfection 1670](#)
5. [Настройка Wake-on-Lan](#)

9. [**Справка по основным командам Gentoo/Calculate**](#)

1. [portage](#) - система управления пакетами в Gentoo
2. [eix](#) - набор утилит для поиска eбилдов по дереву Portage и получения информации о них, в том числе работа с локальными настройками, внешними оверлеями, версиями пакетов и др.
3. [layman](#) - утилита для управления оверлеями Gentoo
4. [opengc](#) - система управления службами системы, запуском и завершением работы хоста
5. [portage-utils](#) - набор легких и быстрых утилит, написанных на C, для извлечения информации о пакетах в Portage
6. [gentoolkit](#) - набор скриптов для администрирования систем, работающих на Gentoo
7. [gentoolkit-dev](#) - набор скриптов в помощь разработчикам под Gentoo

10. [**Руководства**](#)

1. [Описание IRC](#)
2. [Git распределённая система управления версиями файлов, правка последнего коммита](#)
3. [Руководство по оформлению программ на Python](#)
4. [Программный Raid](#)
5. [Руководство по Iptables](#)
6. [Перекодировка тр3 тегов](#)

Содержание этого документа распространяется на условиях [Creative Commons - Attribution / Share Alike](#) лицензии.

1. Установка Calculate

1. [Об установке Calculate Linux](#)
2. [Краткое руководство по установке](#)
3. [Установка на жёсткий диск](#)
4. [Установка на Flash](#)
5. [Аппаратные требования](#)
6. [Структура FTP-зеркала](#)
7. [Разбиение диска](#)
8. [Программное обеспечение](#)
9. [Установка с загрузочного CD-диска](#)
10. [Чем заняться дальше?](#)

Об установке Calculate Linux

Добро пожаловать!

Спасибо за Ваш интерес к Calculate Linux. Мы постоянно работаем над [удобством работы](#) с системой и надеемся, что работа с Calculate Linux доставит Вам истинное удовольствие.

Подробнее о [проекте Calculate Linux](#).

Как организована установка?

Calculate Linux распространяется в виде загрузочного Stage4-образа. Во время загрузки с LiveCD и установки системы на компьютер утилиты Calculate настраивают систему при помощи шаблонов. Таким образом, на LiveCD вы имеете точную копию устанавливаемой системы.

Программы установки, запущенной из командной строки, могут быть переданы все опции установки. Существующие пользователи будут перенесены, установка из Stage4 происходит путем распаковки. Тем самым время на установку сокращается до минимума.

Вы легко можете изменить настройки и состав программ, создав измененный ISO-образ или обновив squashfs-файл на USB Flash. В этом случае установка будет выполняться со всеми внесёнными изменениями. Для этого воспользуйтесь [следующим руководством](#).

Какие варианты установки существуют?

Установить систему Calculate Linux вы можете одним из перечисленных способов:

- графическим клиентом [cl-console-gui](#);
- консольным клиентом [cl-console](#);
- напрямую сервером утилит [cl-core](#).

Система может быть установлена из squashfs-образа, если Вы загрузились с liveCD или USB Flash, либо из ISO-файла, размещенного в директории `/var/calculate/linux` или `/var/calculate/remote/linux`. Во втором случае Вы можете установить любую версию дистрибутива поддерживаемой архитектуры.

Сервер утилит, консольный и графический клиенты входят в состав Calculate Linux, но могут быть установлены из оверлея Calculate в любом Gentoo-совместимом дистрибутиве.

Появились затруднения?

Если у вас есть вопросы, не стесняйтесь задавать их в [IRC-чате](#), посвящённом Calculate Linux. Канал `#calculate-ru` находится на сервере Freenode, а в качестве IRC клиента Вы можете использовать программы `konversation` или `hexchat`, [которые входят](#) в поставку Calculate Linux Desktop.

Вы также можете обратиться к нашему сообществу через [списки рассылки](#).

Найти единомышленников можно в социальной сети [ВКонтакте](#), [Facebook](#) или [Twitter](#).

Краткое руководство по установке

Благодарим за использование Calculate Linux!

Мы постарались сделать для вас максимально удобную для работы систему, используя оригинальный установщик, переработанный интерфейс, шаблоны настройки, утилиты Calculate и Gentoo Portage. Дистрибутив распространяется в виде установочного образа, содержащего лучшее программное обеспечение. Большая часть программ имеет свободную лицензию, позволяющую не только устанавливать и распространять, но и модифицировать исходный код. Используемые сокращенные названия дистрибутивов:

- CLD - Calculate Linux Desktop KDE
- CLDM - Calculate Linux Desktop MATE
- CLDX - Calculate Linux Desktop XFCE
- CLS - Calculate Linux Scratch
- CMC - Calculate Media Center
- CDS - Calculate Directory Server
- CSS - Calculate Scratch Server

Для получения прав администратора системы, запущенной с LiveCD, либо находясь в графическом режиме, выполните su в виртуальном терминале, либо перейдите в одну из текстовых консолей нажатием **Ctrl+Alt+F[1-8]**. Доступ к рабочему столу CLD, CLDM и CLDX выполняется пользователем **guest** с паролем **guest**.

Настройка сети

Настройка сети в Calculate Linux осуществляется с помощью сервера утилит Calculate. Как и все действия сервера утилит, настройку сети можно выполнить несколькими способами:

- используя графический клиент;
- используя консольный клиент;
- используя сервер утилит.

Подробнее см. в разделе [Настройка сети](#).

Разбивка диска

Перед установкой вам может понадобиться изменить разделы жесткого диска. Для установки CLD, CLDM и CLDX мы рекомендуем использовать раздел не менее 10 Гб. Более подробно аппаратные требования можно узнать [здесь](#). Раздел подкачки (swap), как правило, выделяется вдвое большим размера оперативной памяти. Если раздел под swap уже существует, система будет использовать его. Полезно иметь отдельный раздел для личных файлов (/home). Наши рекомендации по разбиению диска описаны [здесь](#).

В CLD, CLDM и CLDX для изменения разделов диска используется программа *Gparted*. В CDS, CSS и CLS входят только консольные утилиты: *fdisk*, *gdisk* или *cfdisk*.

Чтобы просмотреть список существующих разделов, откройте консоль с правами пользователя *root* и наберите:

```
fdisk -l
```

Примечание: для получения прав пользователя *root* в консоли используйте команду *su* либо *sudo*.

Если вы никогда не использовали Linux, то вам понадобится немного времени на то, чтобы привыкнуть к другому наименованию разделов.

В Linux разделы обозначаются как *sda1*, *sda2*, ... вместо привычных *C:*, *D:*, ...

Вы также можете воспользоваться уже готовым разделом либо создать его из *Windows*. Чтобы правильно определить выбранный раздел в Linux, запомните очерёдность его расположения и размер. Как правило, диску *C:* соответствует *sda1*, диску *D:* - *sda2*.

Варианты установки

Установить систему Calculate Linux вы можете одним из перечисленных способов:

- графическим клиентом [cl-console-gui](#),
- консольным клиентом [cl-console](#),
- напрямую сервером утилитой [cl-core](#).

Система может быть установлена из squashfs-образа, если Вы загрузились с liveCD или USB Flash, либо из ISO-файла, размещенного в директории `/var/calculate/linux` или `/var/calculate/remote/linux`. Во втором случае Вы можете установить любую версию дистрибутива поддерживаемой архитектуры.

Сервер утилит, консольный и графический клиенты входят в состав Calculate Linux, но могут быть установлены из оверлея Calculate в любом Gentoo-совместимом дистрибутиве. Подробное описание установки смотрите в соответствующих разделах:

- [Установка на жёсткий диск](#)
- [Установка на Flash](#)

Первый запуск

Если вы не указали других пользователей, после установки CLD, CLDM и CLDX в системе будут заведены две учётные записи, `root` и `guest`. Доступ к графическому сеансу может получить любой пользователь, кроме `root`. По умолчанию у пользователя `guest` установлен пароль `guest`.

После установки в CLS нет графического приглашения к вводу пароля. Для запуска оконного менеджера выполните:

```
startx
```

Для получения прав пользователя `root` используйте команду `su`. Добавление новых пользователей подробно описано [здесь](#).

По умолчанию вы можете зайти в систему удалённо (по протоколу ssh) только как пользователь `root`. В файле `/etc/ssh/sshd_config` в значение параметра `AllowUsers` можно добавить другие учётные записи. Мы рекомендуем убрать права удалённого доступа к системе для пользователя `root`.

Обновление

Calculate Linux использует модель обновлений rolling release. Вы можете обновлять систему практически неограниченное количество раз, используя утилиту обновления системы [cl-update](#).

Для выполнения синхронизации списка пакетов и обновления программ выполните:

```
cl-update
```

Если вы хотите только обновить список пакетов, то выполните:

```
cl-update --sync-only
```

После этого вы можете установить новые программы при помощи менеджера пакетов `emerge`. Краткая справка приведена [здесь](#). Ознакомьтесь также с [Руководством по обновлению системы](#).

Помощь

Если установка системы вызвала сложности или если вы хотите поделиться своим впечатлением, зайдите на IRC канал `#calculate-ru` (сервер FreeNode) сообщества пользователей Calculate Linux. Для этого достаточно воспользоваться иконкой `Hexchat` на вашем рабочем столе.

Сайт проекта: <http://www.calculate-linux.ru>

IRC чат: <http://www.calculate-linux.ru/irc>

Найти единомышленников можно и в социальных сетях:
[ВКонтакте](#), [Facebook](#), [Google+](#), [Одноклассники](#) или [Twitter](#).

Приятной работы!
Команда разработчиков *Calculate Linux*.

Установка Calculate Linux на жёсткий диск

- [Установка Calculate Linux на жёсткий диск](#)
- [Графическая установка системы](#)
- [Язык и локаль](#)
- [Выбор дистрибутива](#)
- [Распределение места на диске](#)
- [Точки монтирования](#)
- [Сетевые настройки](#)
- [Пользователи](#)
- [Видео](#)
- [Начать установку](#)
- [Установка системы из консоли](#)
- [Помощь](#)

Графическая установка системы

Для установки системы с помощью [графического клиента](#) утилит Calculate выполните одно из действий:

- запустите `c1-console-gui` и в категории **Установка** выберите пункт **Установка системы**;
- запустите `c1-console-gui --method install` для открытия отдельного окна с установкой системы.

Необходимые ярлыки запуска программы можно найти на рабочем столе LiveDVD или в меню.

Установка Calculate Linux состоит из следующих шагов:

1. Язык и локаль;
2. Выбор дистрибутива;
3. Распределение места на диске;
4. Точки монтирования;
5. Сетевые настройки;
6. Пользователи;
7. Настройка видео;
8. Установка на жёсткий диск.

Пользователь может перемещаться по шагам с помощью кнопок *Назад* и *Далее*, а также выбрав нужный пункт в левом меню. Во втором случае, если пропущен обязательный шаг или где-то была допущена ошибка, то перед установкой пользователю будет предложено исправить ошибку.

Язык и локаль

Первым пунктом установки является выбор языка и часового пояса. Выберите необходимые параметры из выпадающих списков.

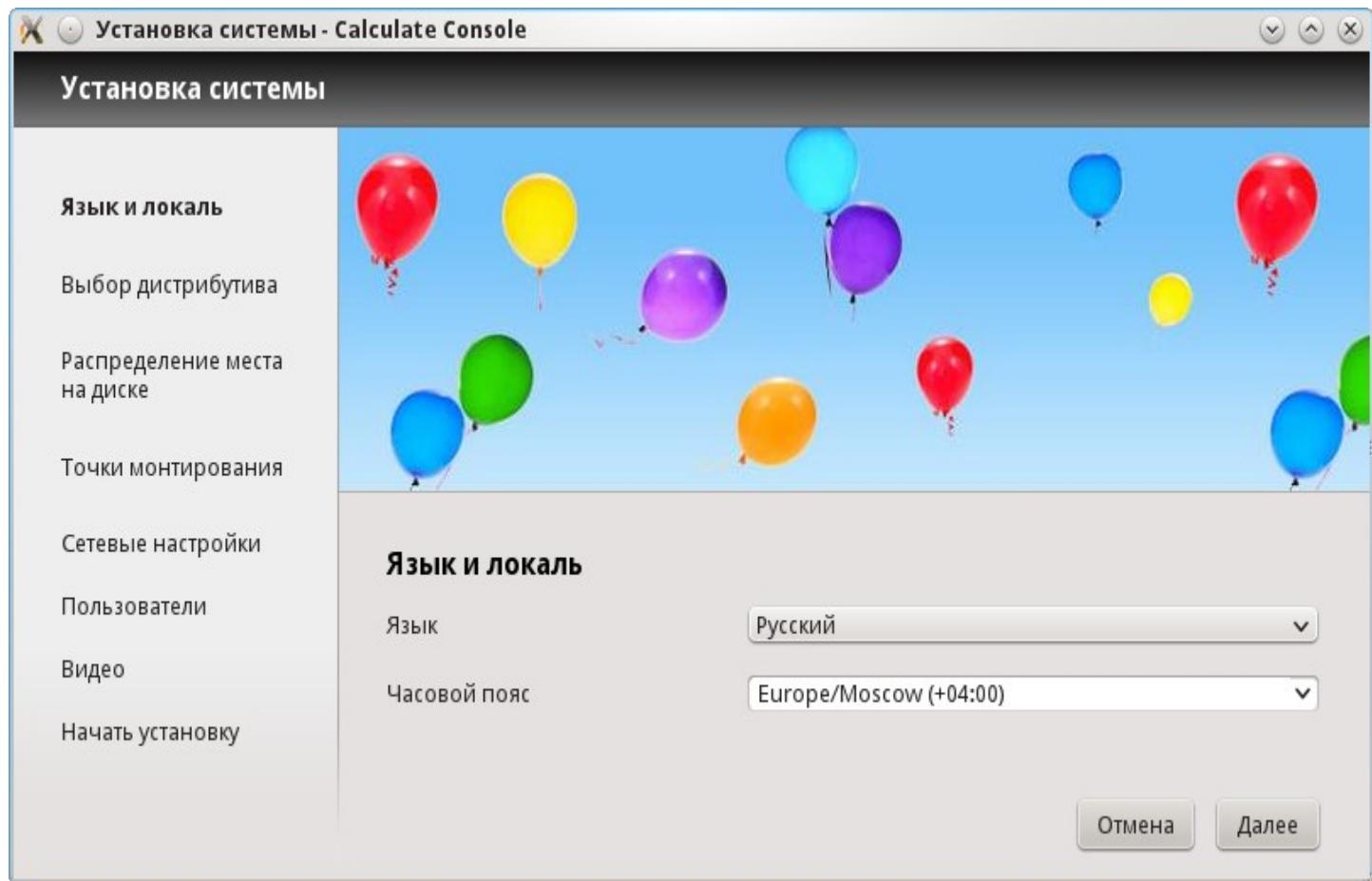


Рис. 1 Настройка языка и локали

Выбор дистрибутива

По умолчанию для установки будет использоваться дистрибутив, с которого был записан ваш DVD или USB-Flash. Программа установки сканирует директории `/var/calculate/linux` и `/var/calculate/remote/linux` и в случае нахождения там ISO образов с Calculate Linux, отобразит их в списке **Установочный образ**.

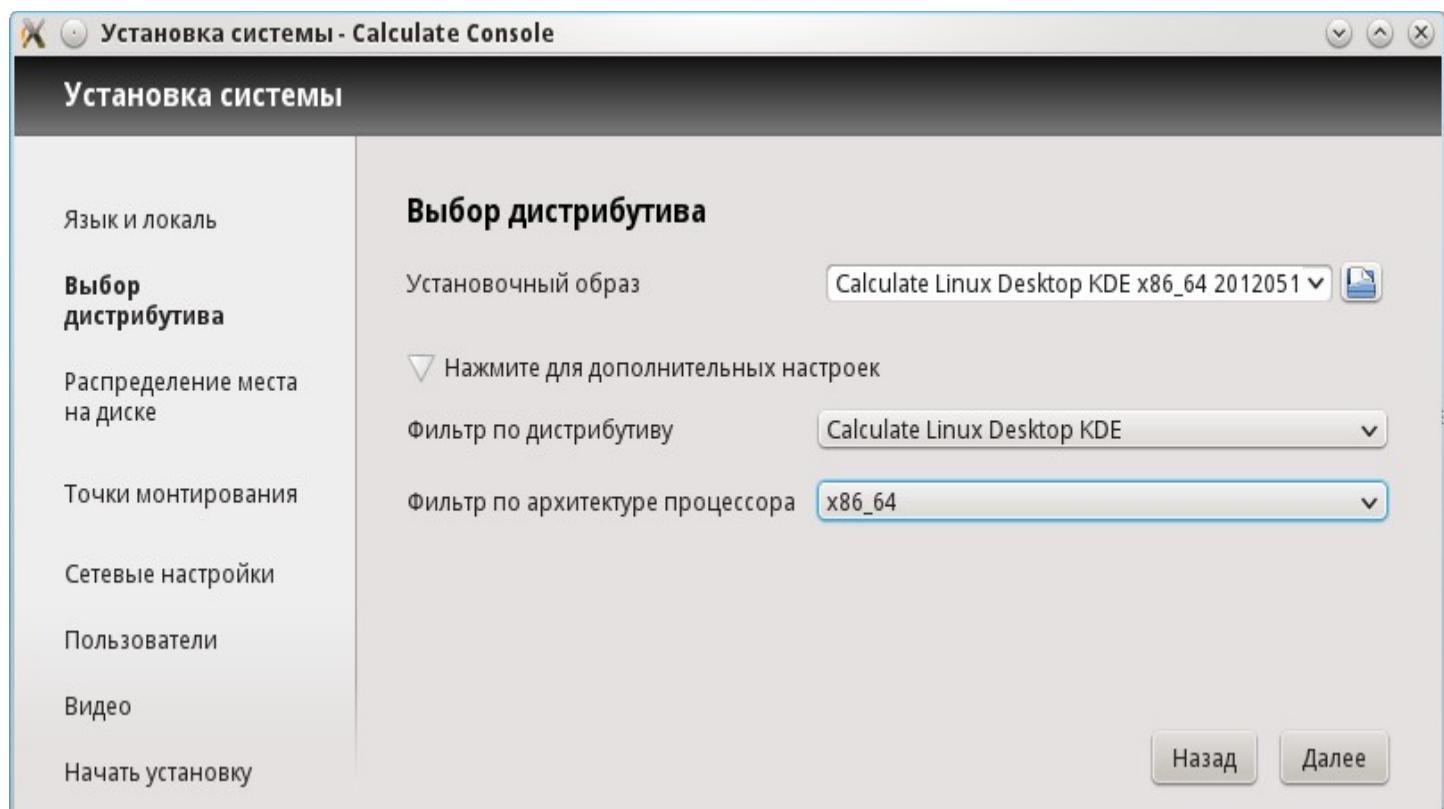


Рис. 2 Выбор дистрибутива

Если у вас есть другие дистрибутивы Calculate Linux, вы можете использовать их для установки, указав путь к файлу образа. Среди дополнительных параметров доступны фильтр по дистрибутиву и фильтр по архитектуре процессора.

Распределение места на диске

В данной версии программы установки вы не можете изменить существующие разделы. Для этого, в зависимости от того, какой дистрибутив вы используете, можно воспользоваться программой GParted или PartitionManager, либо cfdisk, fdisk или gdisk, работающими из консоли во всех версиях Calculate Linux.

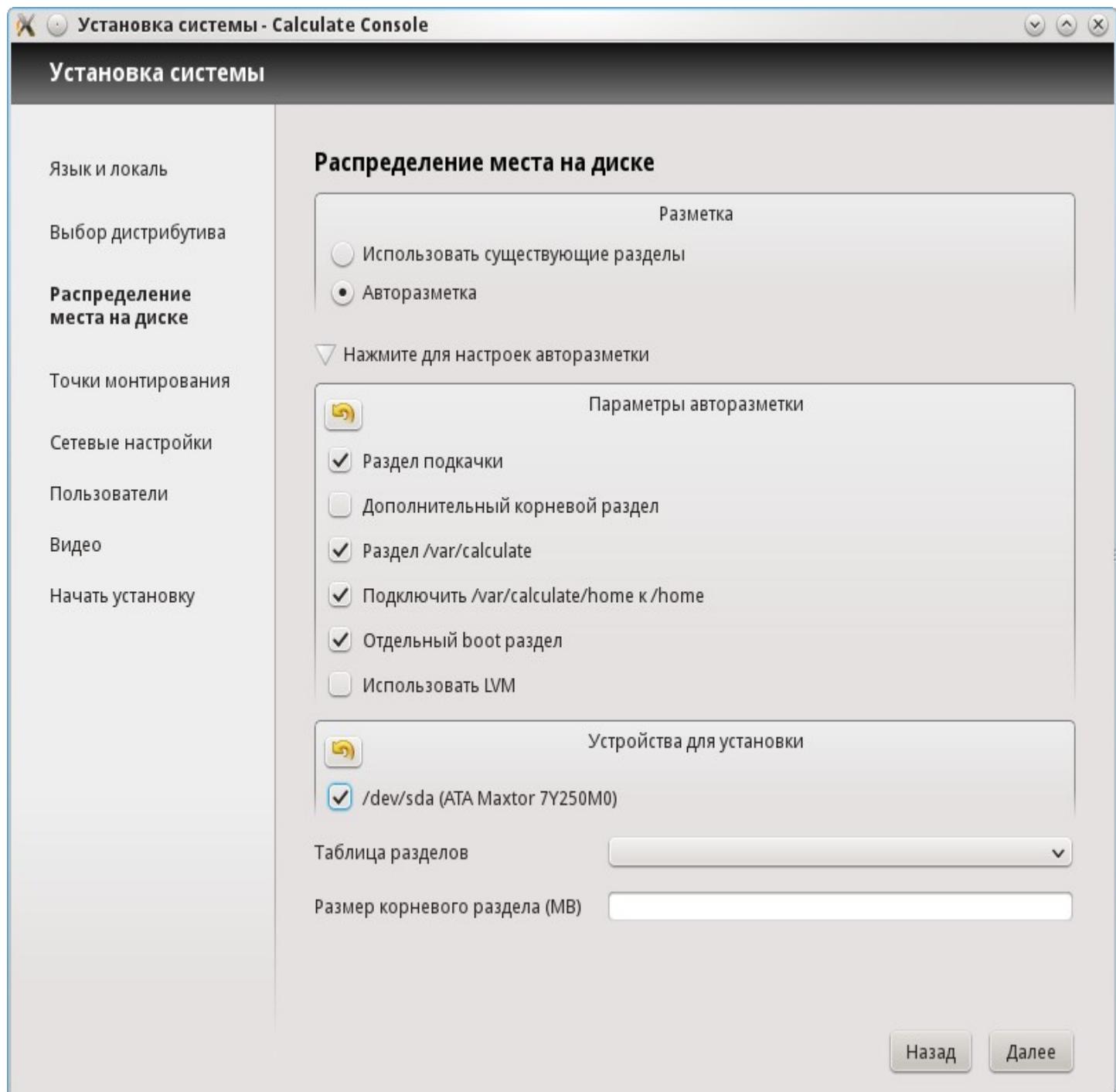


Рис. 3 Распределение места на диске

Если вы готовы использовать весь жёсткий диск под систему, воспользуйтесь авторазметкой, либо выберите пункт "использовать существующие разделы" для указания одного или нескольких разделов жёсткого диска для установки. При выборе авторазметки вы можете воспользоваться дополнительными параметрами:

- *Параметры авторазметки* - здесь вы можете указать, на сколько разделов будет разбит ваш жёсткий диск;
- *Устройства для установки* - выбор устройств для установки;
- *Таблица разделов* - выбор типа таблицы разделов ("DOS-type Partition Table" или "GUID Partition Table (GPT)");
- *Размер корневого раздела* - размер корневого раздела в мегабайтах (размеру 20 Гб будет соответствовать значение 20480).

Точки монтирования

При выборе авторазметки на шаге *Распределение места на диске* данный шаг будет **недоступен для редактирования**.

Основным параметром является выбор точек монтирование в виде таблицы "*Разметка*":

- для изменения значения в строке нажмите на неё и в открывшемся окне введите необходимые значения;
- для добавления новой точки монтирования нажмите на кнопку "+" (плюс) над таблицей;
- для удаления точек монтирования установите флагки слева от необходимых строк и нажмите на кнопку "-" (минус) над таблицей;
- для восстановления первоначальных значений ячеек таблицы нажмите на третью кнопку "возврата" (стрелка) над таблицей;
- для очистки всей таблицы нажмите на четвёртую кнопку "очистки" над таблицей.

В столбце "*Диск или директория*" указывается диск (директория) для монтирования. Во втором столбце "*Точка монтирования*" указывается куда этот диск (директория) будут примонтированы. В третьем столбце "*Файловая система*" указывается, какую файловую систему необходимо использовать для данного диска. В столбце "*Форматировать*" указывается, следует ли форматировать диск. Если указанная файловая система не совпадает с текущей, то диск будет отформатирован. В последнем столбце "*Размер*" указан размер выбранного диска (директории). Среди дополнительных параметров можно:

- установить режим сборки;
- установить использование UUID-дисков;
- выбрать тип установки (Жёсткий диск, USB Flash или USB жёсткий диск);
- выбрать загрузочный диск;
- выбрать I/O планировщик (Deadline, CFQ, No-op).

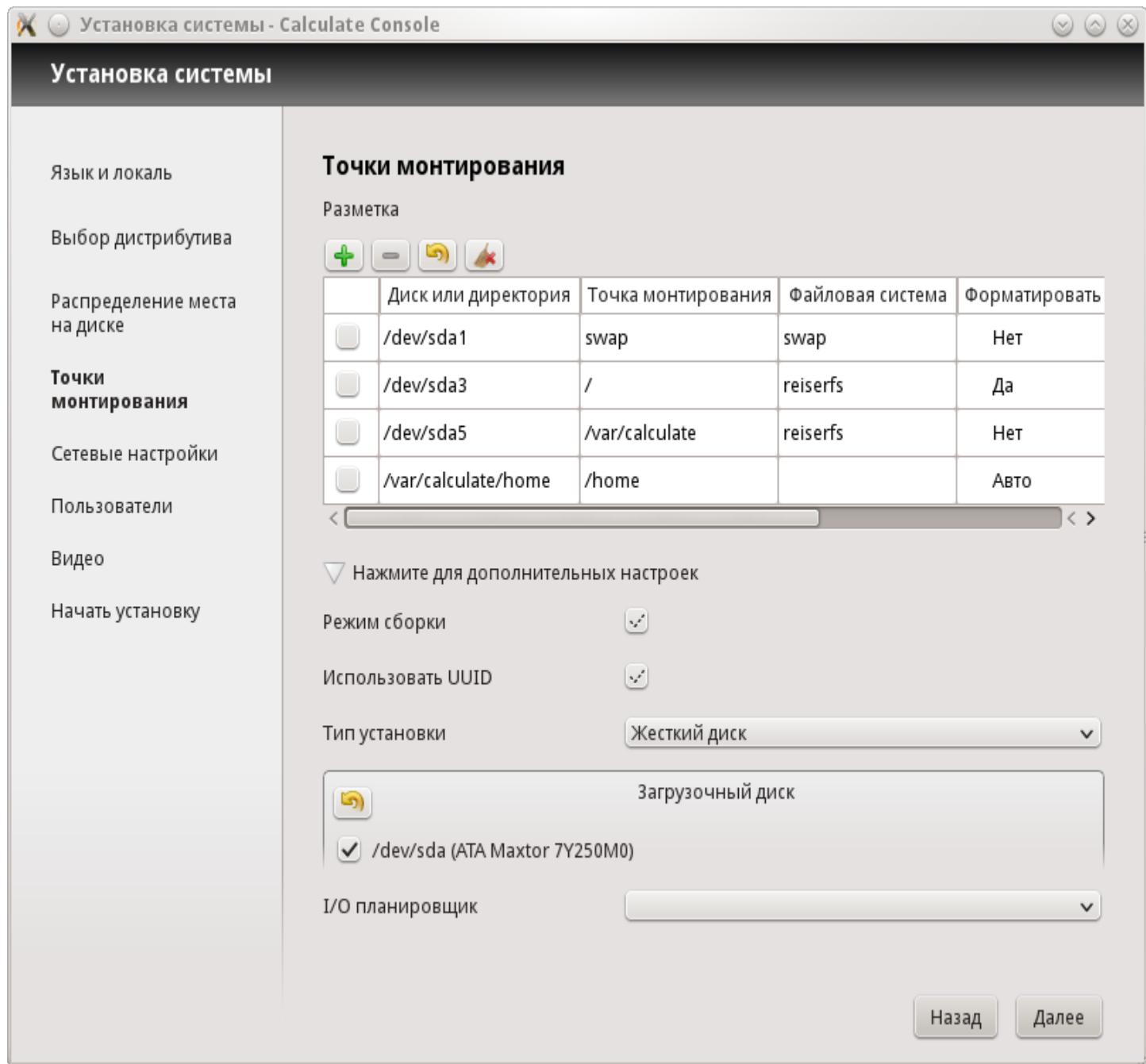


Рис. 4 Выбор точек монтирования

Сетевые настройки

Для настроек сети необходимо:

- выбрать менеджер сети (NetworkManager или OpenRC);
- настроить адреса для интерфейсов в виде таблицы со столбцами:
 1. использование DHCP;
 2. IP-адрес для сетевого интерфейса;
 3. маска сети;
- ввести имя хоста (короткое или полное);
- ввести сервер времени. Также с помощью дополнительных параметров можно указать:
 - сервер доменных имён;
 - домены для поиска;
- таблицу маршрутизации в виде таблицы со столбцами:
 1. сеть;
 2. шлюз;

3. интерфейс;
4. исходный IP.

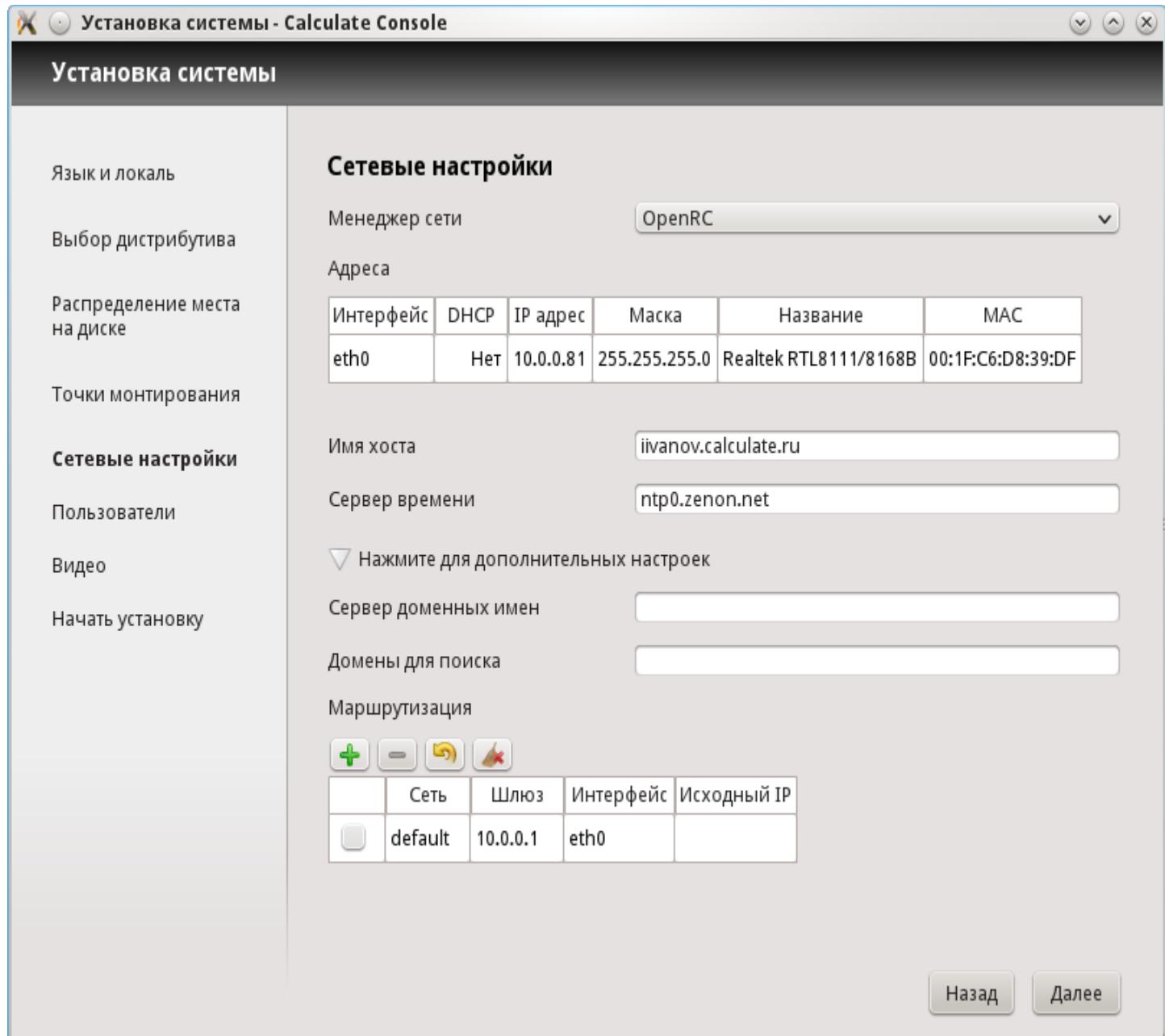


Рис. 5 Сетевые настройки

Пользователи

Настройка пользователей осуществляется изменением таблицы "Переносимые пользователи" со столбцами "Логин" и "Пароль". Работа с таблицей "Переносимые пользователи" аналогична работе с таблицей "Разметка" на шаге "Точки монтирования", рассмотренной выше.

Особенностью является ввод пароля для пользователей, который требует повторного ввода и не отображается в явном виде.

Также из выпадающего списка можно выбрать пользователя для автovхода.

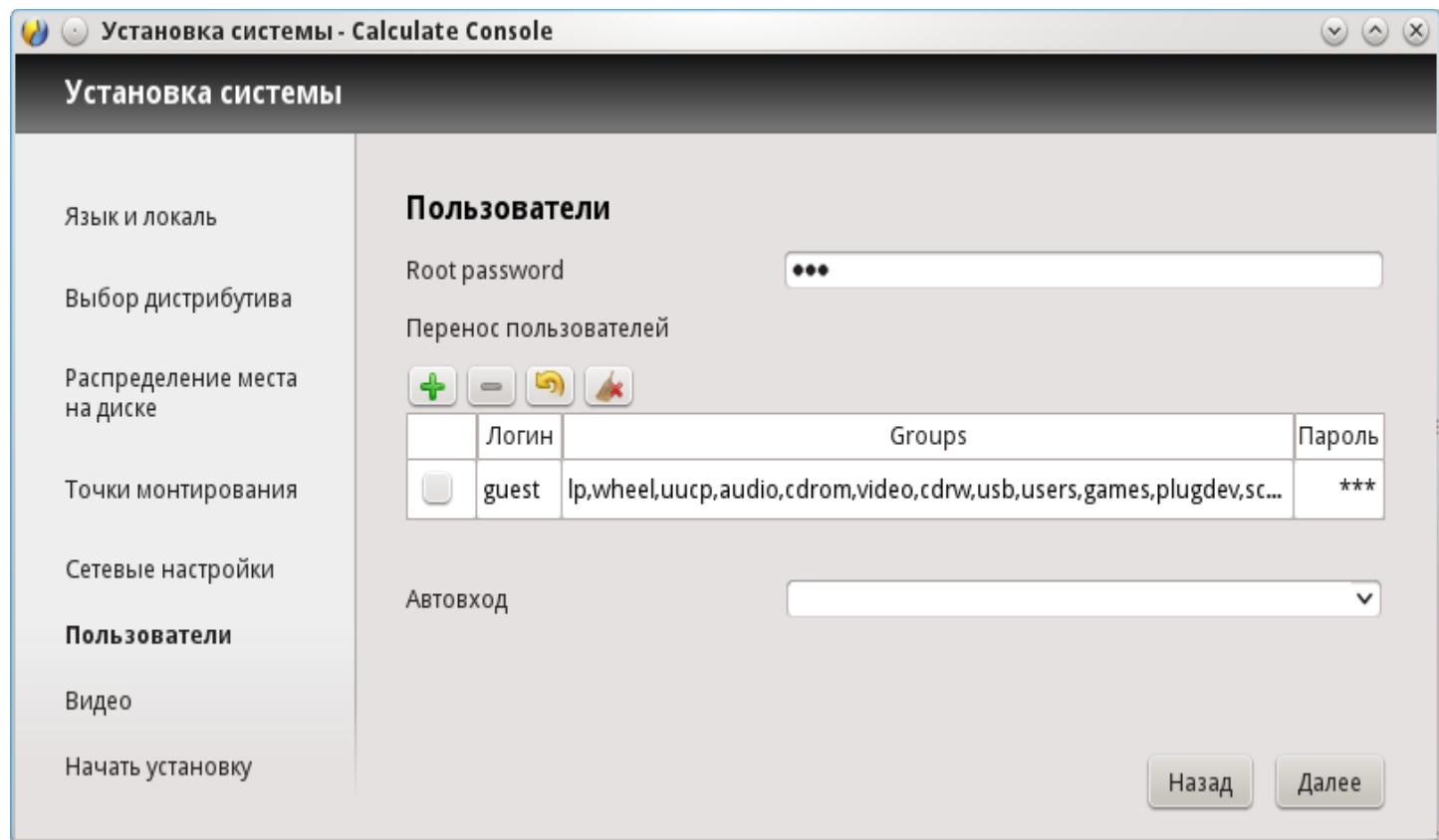


Рис. 6 Настройка пользователей

Видео

Настройка видео состоит из четырёх параметров:

1. Выбор видеодрайвера Xorg из списка (если необходимый драйвер отсутствует в списке, впишите его название);
2. Включение композита;
3. Установка разрешения экрана (если необходимое значение отсутствует в списке, впишите его);
4. Установка разрешения фреймбуфера (если необходимое значение отсутствует в списке, впишите его);

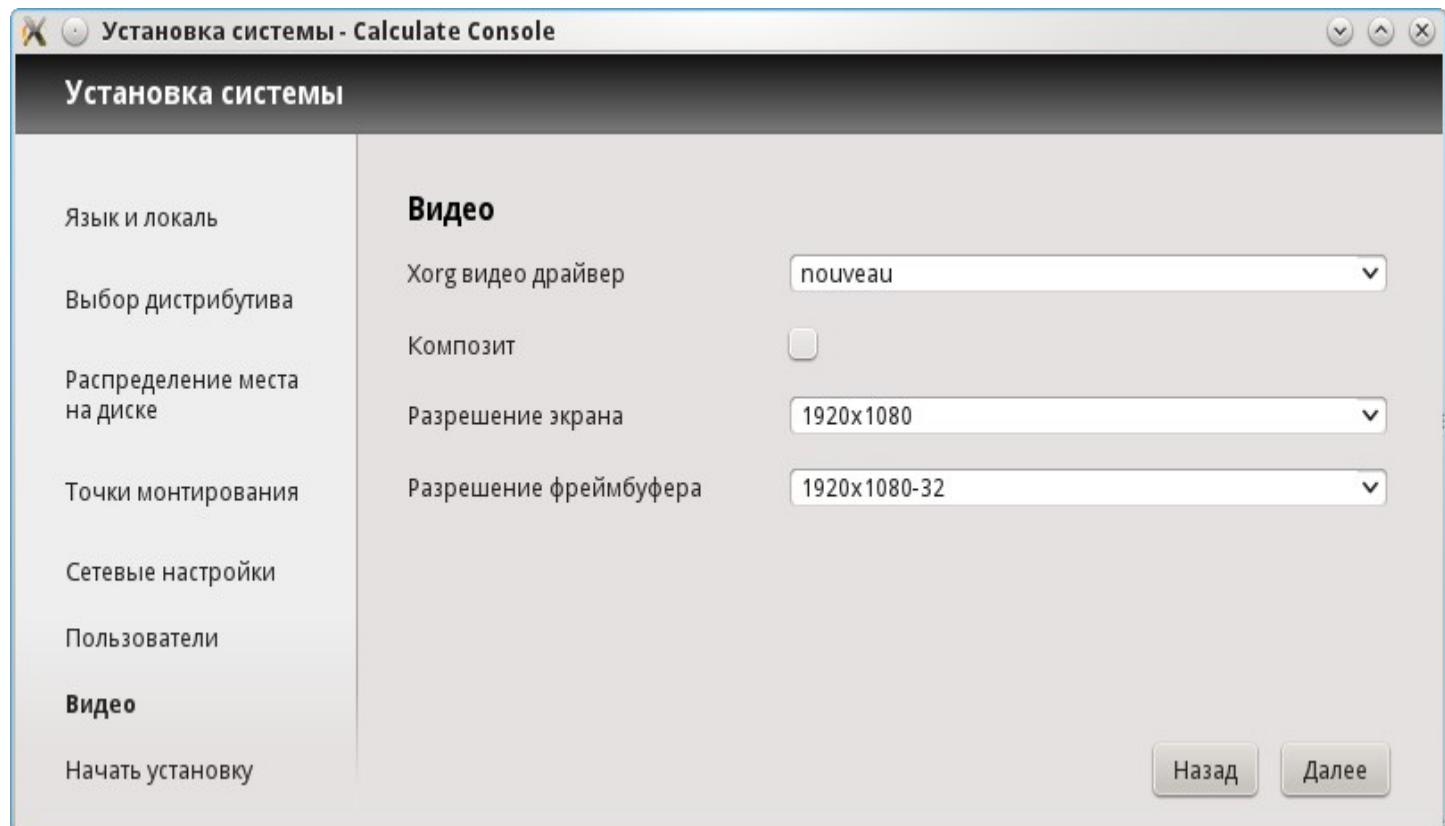


Рис. 7 Настройка видео

Начать установку

Последний шаг является информационным и отображает значения всех параметров, которые будут использоваться при установке.

Для начала установки нажмите на кнопку "Установка".

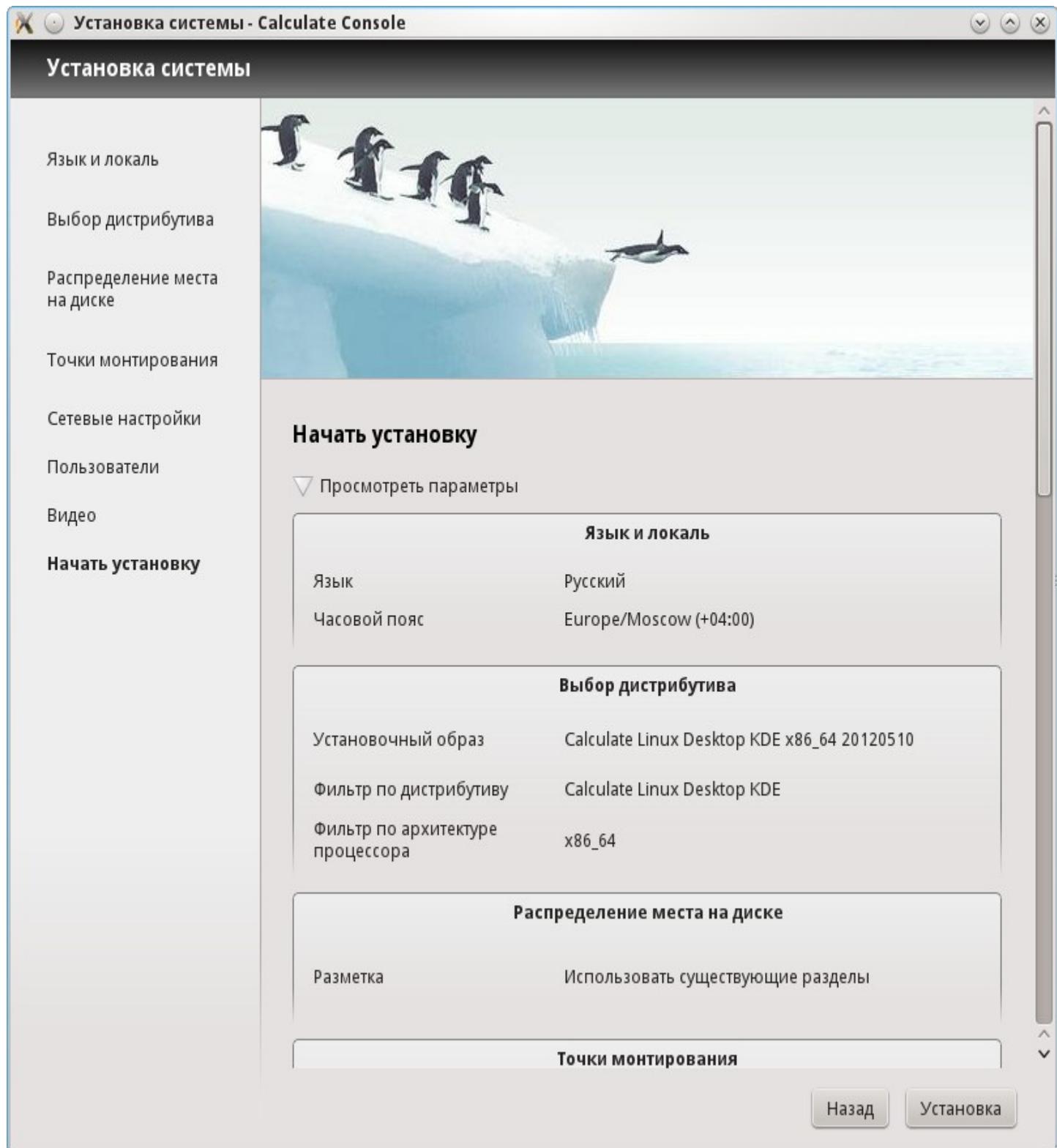


Рис. 8 Просмотр параметров и начало установки

Установка системы из консоли

Для установки системы из консоли используйте команду:
`cl-install`

`cl-install` представляет собой символьическую ссылку на метод [сервера утилит](#):
`cl-core --method install`

Символические ссылки на методы можно создать с помощью команды `cl-core --create-symlink`.

В этом случае система предложит установить систему с параметрами по умолчанию. Для просмотра всех доступных параметров установки системы используйте команду
`cl-install --help`

Все доступные параметры разделены на группы (дублируют шаги в графическом клиенте):

- Язык и локаль
- Выбор дистрибутива
- Распределение места на диске
- Точки монтирования
- Сетевые настройки
- Пользователи
- Видео

Язык и локаль

Доступные параметры:

- `-l LANG, --lang LANG` - установка языка
- `--timezone TIMEZONE` - установка часового пояса

Пример:

```
cl-install -l ru_RU --timezone Europe/Moscow
```

Выбор дистрибутива

Доступные параметры:

- `--iso IMAGE` - ISO образ для установки
- `-s SYSTEM, --os SYSTEM` - выбор операционной системы (CDS,CLDG,CLD,CLDX,CLS,CMC,CSS или Gentoo)
- `--march ARCH` - выбор архитектуры процессора (auto,i686 или x86_64)

Пример:

```
cl-install --iso /var/calculate/linux/cld-20110904-x86_64.iso
cl-install -s CLDX --march i686
```

В первом примере выбран конкретный образ дистрибутива, а во втором сервер утилит будет искать 32-битный образ CLDX среди доступных.

Распределение места на диске

Доступные параметры:

- `--autopartition` - использовать авторазметку;
- `--auto-scheme AUTOPARTOPTS` - параметры авторазметки (swap,root,data,home,boot,uefi,grub или lvm);
- `-D DEVICE` - установить диск для авторазметки;
- `--partition-table TABLE` - установить таблицу разделов для авторазметки (dos или gpt);
- `--root-size SIZE` - установить размер корневого раздела для авторазметки.
- `--swap-size SIZE` - установить размер раздела подкачки для авторазметки.

Параметр `--auto-scheme`, используемый при авторазметке, может принимать одно или несколько значений из swap,root,data,home,boot,lvm,uefi,grub, где

swap - использовать раздел подкачки;

root - использовать дополнительный корневой раздел;

data - использовать раздел /var/calculate;

home - подключить /var/calculate/home к /home;
boot - использовать отдельный boot раздел;
grub - создать bios_grub раздел;
uefi - использовать UEFI загрузчик;
lvm - использовать LVM.

Примеры:

```
cl-install --autopartition on --auto-scheme swap,root,data,home -D  
/dev/sda
```

Для установки будет использован диск /dev/sda с авторазметкой, будут использованы разделы подкачки, /var/calculate, дополнительный корневой раздел и подключён /var/calculate/home к /home.

```
cl-install --autopartition --partition-table dos --root-size 10240
```

При установке будет использована авторазметка, таблица разделов для авторазметки установится в значение dos и размер корневого раздела в 10 Гигабайт.

Точки монтирования

Доступные параметры:

- -d DISKS, --disk DISKS - установка точки монтирования;
- --build [ON/OFF] - установка для сборки;
- --uuid [ON/OFF] - использовать UUID;
- --type DISKTYPE - тип устройства для устанавливаемой системы (hdd, flash или usb-hdd);
- --mbr MBR - загрузочный диск для устанавливаемой системы;
- --scheduler SCHEDULER - установить I/O планировщик (deadline, cfq или noop).
- --uefi [ON/OFF] - использовать загрузку UEFI

Пример:

```
cl-install -d /dev/sda2::ext4:on -d /dev/sda3:/var/calculate:reiserfs  
-d /dev/sda1:swap --scheduler cfq
```

Диск /dev/sda2 (первый параметр после ключа -d) будет примонтирован в корень (второй параметр) и будет обязательно отформатирован (четвёртый параметр "он") в файловую систему ext4 (третий параметр).

Диск /dev/sda3 будет примонтирован в /var/calculate и отформатирован в файловую систему reiserfs, только если в текущий момент имеет другую файловую систему; если текущая файловая система reiserfs, то формироваться не будет.

Диск /dev/sda1 будет примонтирован для swap.

Будет использован I/O планировщик cfq.

Сетевые настройки

Доступные параметры:

- --netconf NETMANAGER - выбор менеджера сети (networkmanager или openrc)
- --iface IFACE_SETTINGS - установка адреса для сетевого интерфейса
- --hostname HOSTNAME - установка короткого или полного имени хоста
- --ntp NTP - установка NTP сервера для системы
- --dns DNS - установка серверов доменных имен (разделитель - запятая)
- --domain-search DOMAINS - установка доменов для поиска (разделитель - запятая)
- --route NETROUTE - добавить правило маршрутизации (формат NETWORK:[GATEWAY]
[:DEV[:SOURCE]])

Пример:

```
cl-install --netconf openrc --iface eth0:192.168.1.47:24 --hostname iivanov.company.ru --route default:192.168.1.1:eth0:192.168.1.47
```

Данной командой в качестве менеджера сети будет установлен openrc, для сетевого интерфейса eth0 отключится DHCP, установится IP-адрес 192.168.1.47 и маска сети 255.255.255.0. Также будет задано имя хоста iivanov.company.ru и маршрутизация для сетевого интерфейса eth0 по умолчанию через 192.168.1.1.

Пользователи

Доступные параметры:

- `-u USERS, --users USERS` - добавить пользователя в установленную систему
- `-A USER, --autologin USER` - указать пользователя для автовахода в установленную систему
- `@-C [ON/OFF] --crypt-home [ON/OFF]` - шифровать пользовательские профили

Примеры:

```
cl-install -u root -u user1
```

В установленной системе будут пользователи root и user1. Автоваход выполниться не будет.

```
cl-install -u root -u user1 -u guest -A user1
```

В установленной системе будут пользователи root, guest и user1. Автоваход будет установлен для пользователя user1.

Видео

Доступные параметры:

- `--video VIDEODRV` - установить видео драйвер
- `--composite [ON/OFF]` - установить композит
- `-X <width>x<height>` - установить разрешение Xorg
- `--fb <width>x<height>` - установить разрешение фреймбуфера
- `--grub-terminal TERMINAL` - установить grub-терминал (console,gfxterm)

Пример:

```
cl-install --video nouveau -X 1920x1080 --fb 1920x1080-32 --composite
```

В установленной системе будет использованы драйвер nouveau, разрешение Xorg 1920x1080, разрешение фреймбуфера 1920x1080-32 и композит.

Помощь

Если установка системы вызвала сложности или вы хотите поделиться своим впечатлением, зайдите на IRC канал `#calculate-ru` (сервер FreeNode) сообщества пользователей Calculate Linux. Для этого кликните на иконке "Сообщество Calculate Linux" на вашем рабочем столе.

Установка на Flash

- [Установка на Flash](#)
- [Установка системы на Flash графическим клиентом](#)
- [Установка системы на Flash консольным клиентом](#)
- [Установка системы на Flash с помощью сервера утилит](#)

Прежде чем приступить к установке, сделайте резервную копию своих данных на флешке.

Подключите флешку к вашему компьютеру. Для определения имени устройства вашей флешки выполните в консоли с правами *root*:

```
fdisk -l
```

Например, если ваш флаш-диск определился как устройство */dev/sdb*, установка системы на него производится командой:

```
cl-install -d /dev/sdb1
```

Установка системы на Flash графическим клиентом

Для установки системы на Flash с помощью [графического клиента утилит Calculate](#) выполните одно из действий:

- запустите *cl-console-gui* и в категории **Установка** выберите пункт **Установка на Flash**;
- запустите *cl-console-gui --method install_flash* для открытия окна с установкой системы. Для установки Calculate Linux на Flash необходимо указать лишь небольшое количество параметров:
- Диск для установки. Как правило, определяется автоматически, иначе необходимо выбрать его из выпадающего списка;
- Установочный образ;
- Разметку диска, которая является необязательным параметром.

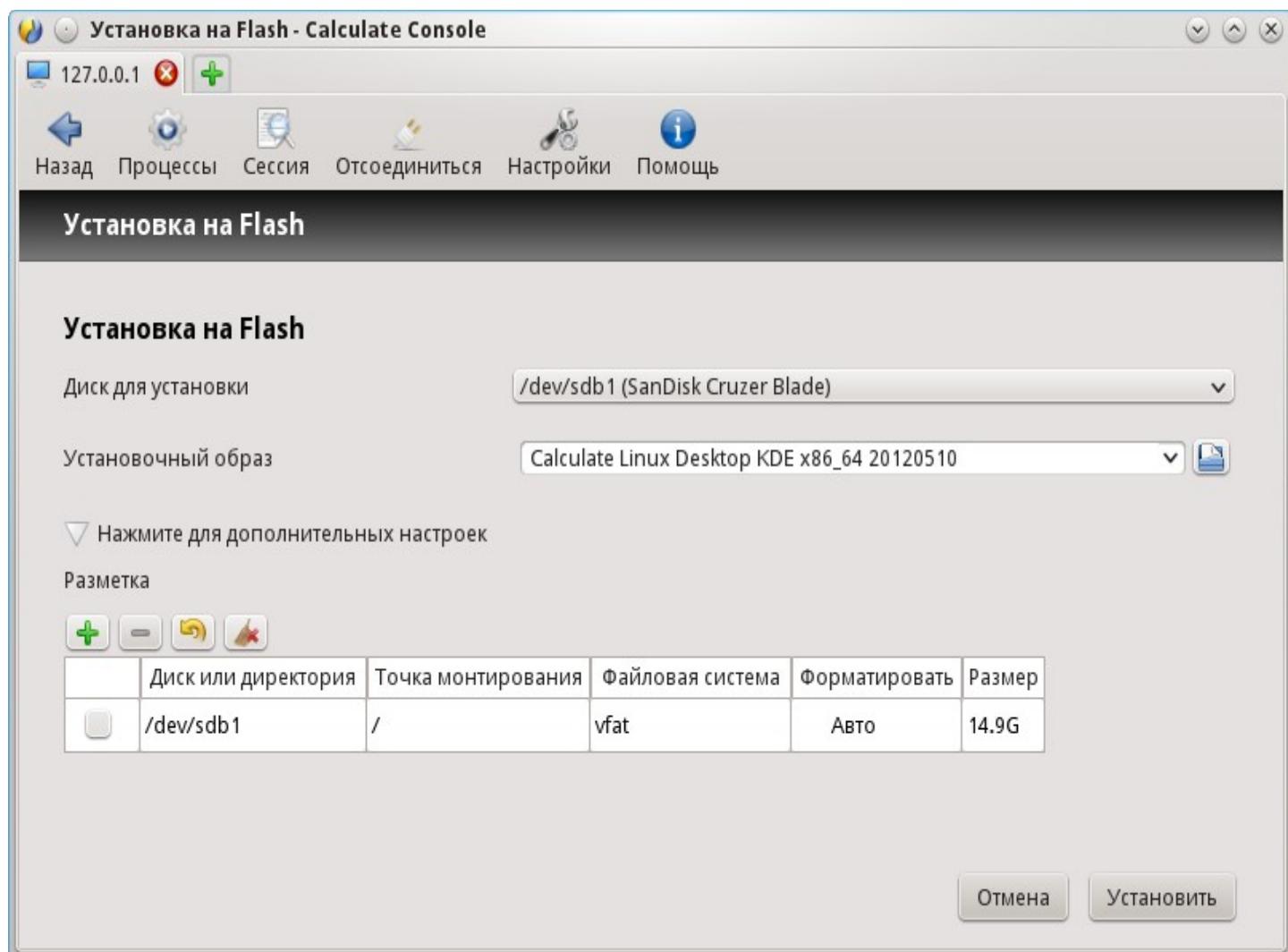


Рис. 1 Установка Calculate Linux на Flash

Установка системы на Flash консольным клиентом

Установка системы на Flash консольным клиентом ничем не отличается от [установки системы на жёсткий диск](#), за исключением меньшего количества используемых параметров.

Для указания раздела для установки используется ключ `-d, --disk`.

Для указания образа дистрибутива для установки используется ключ `--iso`. Для установки системы не обязательно указывать образ дистрибутива при нахождении его в директориях по умолчанию: `/var/calculate/linux` и `/var/calculate/remote/linux`.

Например:

```
cl-console --method install -d /dev/sdb1
```

Пример установки системы на Flash с указанием файла образа дистрибутива:

```
cl-console --method install -d /dev/sdb1 --iso /var/calculate/linux/cld-20110904-x86_64.iso
```

Установка системы на Flash с помощью сервера утилит

Установка системы на Flash с помощью сервера утилит ничем не отличается от [установки системы на жёсткий диск](#), за исключением меньшего количества используемых параметров.

Для указания раздела для установки используется ключ `-d, --disk`.

Для указания образа дистрибутива для установки используется ключ `--iso`. Для установки системы не обязательно указывать образ дистрибутива при нахождении его в директориях по умолчанию: `/var/calculate/linux` и `/var/calculate/remote/linux`.

Для установки используется команда `cl-core --method install`:

```
cl-core --method install -d /dev/sdb1
```

или символьическая ссылка на неё `cl-install`, например

```
cl-install -d /dev/sdb1 --iso /var/calculate/linux/cld-20110904-x86_64.iso
```

Аппаратные требования

Минимальные требования к аппаратному обеспечению

Проверьте, соответствует ли ваше аппаратное обеспечение требованиям выбранной версии системы.

У вас может оказаться компьютер меньшей мощности, чем рекомендовано в таблице ниже. При этом установка вполне возможна, однако есть вероятность остаться неудовлетворённым работой системы - поэтому следует учитывать наши рекомендации.

Возможно запускать графическое окружение рабочего стола на старых или дешёвых машинах. В этом случае рекомендуется установить менеджер окон, менее требовательный к ресурсам по сравнению с KDE, - например, XFCE.

Calculate Directory Server

Центральный процессор i686 или новее (Intel не ниже Pentium Pro, AMD не ниже Athlon)

Память 256 Мб

Дисковое пространство 4 Гб

Пространство подкачки как правило, не меньше количества оперативной памяти

Calculate Linux Desktop KDE

Центральный процессор i686 или новее (Intel не ниже Pentium Pro, AMD не ниже Athlon)

Память 512 Мб

Дисковое пространство 7 Гб

Пространство подкачки как правило, не меньше количества оперативной памяти

Calculate Linux Desktop Xfce

Центральный процессор i686 или новее (Intel не ниже Pentium Pro, AMD не ниже Athlon)

Память 256 Мб

Дисковое пространство 5 Гб

Пространство подкачки как правило, не меньше количества оперативной памяти

Calculate Linux Scratch

Центральный процессор i686 или новее (Intel не ниже Pentium Pro, AMD не ниже Athlon)

Память 128 Мб

Дисковое пространство 3 Гб

Пространство подкачки как правило, не меньше количества оперативной памяти

Требования к диску и памяти могут возрасти в зависимости от типа выполняемых задач.

Структура FTP-зеркала

FTP-зеркало с дистрибутивами Calculate Linux выглядит примерно следующим образом:

`ftp://ftp.calculate-linux.org/calculate/release/15.12`

где: 'ftp.calculate-linux.org/calculate' - адрес FTP-сервера, 'release' - директория с релизами, '15.12' - версия системы.

Структура файлов в директории выглядит следующим образом:

```
cld-15.12-x86_64.iso  
cld-15.12-x86_64.iso.torrent  
cld-15.12-x86_64.list  
md5sum.txt  
sha1sum.txt
```

где:

- `_cld-15.12-x86_64.iso` - загрузочный Live USB образ системы, с возможностью [установки](#);
- `_cld-15.12-x86_64.iso.torrent` - BitTorrent файл - предпочтительный способ для быстрой загрузки системы и дополнений;
- `_cld-15.12-x86_64.list` - список пакетов, входящих в дистрибутив;
- `md5sum.txt` и `sha1sum.txt` - контрольные суммы файлов.

Примечание:

Для 32-битной архитектуры в названиях пакетов будет обозначение **i686** вместо **x86_64**.

Разбиение диска

Общая схема

[Calculate Linux](#) можно переустановить без каких-либо дополнительных параметров при следующем разбиении диска:

```
/dev/sda1 swap  
/dev/sda2 10-20Gb Linux (/)
```

```
/dev/sda3 10-20Gb Linux (/)  
/dev/sda4 Extended  
/dev/sda5 Linux (/home)  
... (другие диски и разделы)
```

Если система загружена с раздела "sda2", то при переустановке системы установщик предложит "sda3", и наоборот. Программа установки помнит, с какого раздела производилась установка системы, и в дальнейшем предложит выполнить обновление в предыдущий раздел.

В настольной версии системы "swap"-раздел может отсутствовать вовсе при достаточном количестве оперативной памяти (2 Гб и выше). В этом случае ядро практически не будет "свопить", интенсивнее высвобождая память.

Преимущества общей схемы

- **защищённость** - пользовательские данные, размещенные на отдельном разделе, никогда не пострадают при переустановке системы;
- **свободное место** - у вас всегда будет свободное место на диске, т.к. вероятность, что какому-то разделу (например, /boot) не хватит места, сведена к минимуму;
- **обновление системы** - вы сможете выполнять обновление, продолжая работать в системе;
- **надёжность** - вы всегда сможете загрузиться в предыдущую систему, если новая по какой-либо причине будет работать нестабильно.

Настройки

Для подключения дополнительных разделов к системе используйте файл /etc/fstab.

Для монтирования раздела в директорию /home достаточно в конец файла /etc/fstab вписать строку примерно следующего содержания:

```
/dev/sda5 /home ext4 noatime 0 0
```

В приведенном примере раздел sda5 с файловой системой ext4 монтируется в директорию /home.

Для подключения раздела выполните:

```
mount /home
```

Прописанный таким образом раздел будет автоматически монтироваться во время загрузки системы.

При переустановке системы *Calculate* будет переносить точки монтирования из файла /etc/fstab в новую систему.

Разбиение диска для сервера

Для разбиения диска под [Calculate Directory Server](#) мы предлагаем использовать следующую схему:

```
/dev/sda1 swap  
/dev/sda2 20Gb Linux (/, ext4)  
/dev/sda3 20Gb Linux (/, ext4)  
/dev/sda4 Extended  
/dev/sda5 Linux (/var/calculate, xfs)
```

Размер диска подкачки (swap) следует выбрать произвольно, исходя из объема оперативной памяти и предполагаемой нагрузки. Как правило, рекомендуют использовать в 2 раза больше, чем объём оперативной памяти.

По мере необходимости вы можете расширить ёмкость разделов, подключив дополнительные диски:

```
/dev/sdb1 Linux (/var/calculate/server-data, xfs)  
/dev/sdc1 Linux (/var/calculate/server-data/samba/share, xfs)
```

Различные файловые системы имеют свои преимущества и недостатки. Мы можем оставить здесь лишь рекомендацию на основе личного опыта. В качестве файловой системы для корневого раздела хорошо зарекомендовала себя "ext4", а вот для хранения файлов, благодаря активному использованию кэша, идеальной, на наш взгляд, является "xfs".

Разбиение диска для Linux-десктона

Настраивая [Calculate Linux Desktop](#) для работы в качестве клиента сервера [CDS](#), мы рекомендуем разбить жесткий диск следующим образом:

```
/dev/sda1 swap  
/dev/sda2 10Gb Linux (/, ext4)  
/dev/sda3 10Gb Linux (/, ext4)  
/dev/sda4 Extended  
/dev/sda5 Linux (/var/calculate, ext4)
```

Обратите внимание, что свободный раздел монтируется в `/var/calculate`. Таким образом, настройки подключения к серверу сохраняются в отдельном разделе, упрощая переустановку системы.

Содержимое раздела `/home` не будет теряться при переустановке, т.к. после входа в домен CDS, директория `/home` располагается в `/var/calculate/home` (монтируется через *bind*). Сохранять данные имеет смысл только для кэширования.

Переустановка системы

При любом разбиении диска архивы с обновлениями удобно хранить в отдельном разделе - пусть это будет, например, `/home` (если он вынесен на отдельный раздел).

Подключить такой диск можно, используя следующую запись в `/etc/fstab`:

```
/home/calculate /usr/calculate/share none bind 0 0
```

Программное обеспечение

Состав программного обеспечения подобран с учётом трёх правил:

1. программа должна быть востребована,
2. иметь единый стиль с оконным менеджером,
3. не дублировать функционал.

Вы можете присыпать свои пожелания к изменению программ в [список рассылки](#) calculate-user-ru.

Calculate Linux Desktop 14.16.2

Интернет

тип программы KDE XFCE

Браузер [Chromium](#) [Chromium](#)

Почта [KMail](#) [Claws Mail](#)

Jabber/ICQ [Kopete](#) [Pidgin](#)

IRC [Konversation](#) [HexChat](#)

Twitter [Choqok](#)

RSS-reader [Akregator](#) [Liferea](#)

Torrent [KGet](#) [Deluge](#)

Офис

тип программы KDE XFCE

Офис [LibreOffice](#) [LibreOffice](#)

Электронные книги [FBReader](#) [FBReader](#)

Графика

тип программы KDE XFCE

Графический редактор [Gimp](#) [Gimp](#)

Просмотр изображений [Gwenview](#) [Geeqie](#)

Менеджер изображений [digiKam](#) [Shotwell](#)

Сканирование изображений [XSane](#) [XSane](#)

Мультимедиа

тип программы KDE XFCE

Видео-проигрыватель [SMPlayer](#) [SMPlayer](#)

Аудио-проигрыватель [Amarok](#) [Clementine](#)

Запись дисков [K3B](#) [Xfburn](#)

Видеомонтаж [Kdenlive](#)

Система

тип программы KDE XFCE

Файловый менеджер [Dolphin](#) [Thunar](#)

Редактор разделов [KDE Partition Manager](#) [GParted](#)

Установка с загрузочного CD-диска

- [Установка с загрузочного CD-диска](#)
- [Запись образа на диск](#)
- [growisofs](#)
- [K3B](#)
- [Опции загрузки](#)
- [Разбиение диска](#)

Если ваш компьютер имеет CD/DVD-привод, то вы можете воспользоваться установкой с загрузочного Live CD/DVD. Загрузочный диск - это точный прообраз устанавливаемой системы, и благодаря ему вы можете оценить работу дистрибутива непосредственно перед установкой на ваш ПК.

Загрузить образ LiveCD можно [здесь](#), воспользовавшись BitTorrent клиентом либо загрузив файл напрямую с http/ftp-зеркал.

Запись образа на диск

В зависимости от объёма данных для записи вам может понадобиться CD или DVD.

Записывать загруженный образ нужно в «сыром» режиме. Использование программ [growisofs](#) и [K3B](#), входящих в комплект [Calculate Linux Desktop](#), описано ниже.

growisofs

При использовании *growisofs* просто выполните:

```
growisofs -Z /dev/cdrom=/usr/calculate/share/linux/cld-10.0-i686.iso
```

В приведенном примере ISO-файл расположен в директории */usr/calculate/share/linux*.

K3B

При использовании K3B выберите *Tools* » *CD* » *Burn Image* (*Инструменты* » *Компакт-диск* » *Прожечь образ*). Затем в поле *Image to Burn* (*Образ для записи*) укажите ISO-файл и нажмите *Start* (*Запуск*).

Загрузка с установочного диска может потребовать выполнения определенных действий. Уберите все компакт-диски из приводов, перезагрузите компьютер и войдите в *BIOS*. В зависимости от *BIOS*, для этого обычно нужно нажать *DEL*, *F1* или *ESC*. В *BIOS* измените порядок загрузки так, чтобы обращение к *CD-ROM* выполнялось до обращения к жёсткому диску. Этот параметр часто задается в разделе *CMOS Setup*. Если порядок загрузки невозможно изменить, система просто перезагрузится с жёсткого диска, игнорируя *CD-ROM*.

Теперь поместите установочный диск в привод *CD-ROM* и перезагрузитесь. Должно появиться загрузочное приглашение.

Опции загрузки

Перед началом загрузки с CD вы увидите меню загрузки. Выполните необходимые установки, воспользовавшись клавишами **F1-F5** для перехода в меню.

- **F1 Help** - Справка.
- **F2 Language** - Выбор языка. Доступны: русский, украинский, английский, испанский, немецкий, португальский, итальянский, польский и французский.
- **F3 Кеумар** - Выбор раскладки клавиатуры. Если вы выбрали язык в предыдущем пункте, то можете этот пункт пропустить.
- **F4 Timezone** - Часовой пояс. Укажите этот параметр, чтобы время в вашем компьютере совпадало с местным.
- **F5 Video** - Разрешение экрана. В большинстве случаев этот параметр можно не менять.
- **Tab Edit options** - Редактирование строки загрузки ядра. Для опытных пользователей.

Указанные при загрузке *LiveCD* параметры будут использованы в дальнейшем при установке системы.

Разбиение диска

В зависимости от выбранного дистрибутива после загрузки компьютера вы можете воспользоваться одной из программ разбиения диска:

CDS, CLS: *fdisk*, *cfdisk*

CLD: *KDE Partition Manager*, *fdisk*, *cfdisk*

CLDX: *Gnome Partition Editor*, *fdisk*, *cfdisk*

Чем заняться дальше?

- [Чем заняться дальше?](#)
- [Документация](#)
- [Ресурсы Calculate](#)
- [Calculate в интернете](#)

Документация

Примите поздравления! У вас теперь появилась работающая система Calculate. И что же делать дальше? Какие у вас появились возможности? На что стоит взглянуть прежде всего? Calculate даёт своим пользователям богатый выбор возможностей, а следовательно - и множество документированных (или не очень) свойств.

Вам обязательно нужно прочитать следующую часть документации, "[Работа с Calculate](#)", в которой рассказывается, как поддерживать программное обеспечение в актуальном состоянии, как доустанавливать программы. Здесь вы найдёте ответы на часто задаваемые вопросы, узнаете множество полезных команд по управлению системой.

Для управления программным обеспечением Calculate использует [Portage](#). Любой пакет из портежей может быть скомпилирован с учетом ваших требований. Безусловно, было бы неплохо знать, как это всё работает. Вся необходимая документация собрана в разделе "[Работа с Portage](#)". Здесь вы узнаете о USE-флагах, переменных среды, ebuild-файлах.

Узнав основные вещи, вы можете познакомиться с особенностями системы - утилитами Calculate. Вся необходимая документация собрана в разделе "[Утилиты Calculate](#)". Здесь вы научитесь создавать свои шаблоны для настройки системы.

Полный список существующих материалов имеется на [странице документации](#).

Ресурсы Calculate

Естественно, мы всегда рады видеть вас на [форумах Calculate](#), как и на [русскоязычном IRC-канале Calculate](#).

Кроме того, мы можем предложить вашему вниманию несколько [списков рассылки](#), открытых для всех наших пользователей. Сведения о порядке подписки находятся на той же странице.

Calculate в интернете

Вы найдете множество мест, где можете поделиться впечатлениями с другими пользователями. Calculate представлен в социальных сетях [ВКонтакте](#), [Facebook](#) и [Twitter](#).

2. Работа с Calculate

1. [ЧаBO \(FAQ\)](#)
2. [Руководство пользователя](#)
3. [Создание учётных записей](#)
4. [Шифрование домашних директорий](#)
5. [Установка и удаление программ](#)
6. [Руководство по обновлению системы](#)
7. [Сценарии инициализации](#)
8. [Сборка ядра со своей конфигурацией](#)
9. [Интерактивная сборка системы](#)
10. [Системные утилиты](#)
11. [Оптимизация системы](#)
12. [Загрузка модулей ядра](#)
13. [Ревизии системы](#)
14. Пакеты оверлея Calculate: [calcboot](#), [calckernel](#), [calculate-sources](#), [keyexec](#), [pam_keystore](#)

Часто задаваемые вопросы (FAQ)

15. [Часто задаваемые вопросы \(FAQ\)](#)
16. [Языковые настройки и параметры клавиатуры.](#)
17. [Переключение языка](#)
18. [Как поменять язык установленной системы?](#)
19. [Как отключить NumLock при входе в систему?](#)
20. [Настройка звуковой карты](#)
21. [Создание пользователей](#)
22. [Установка программ](#)
23. [Настройка маршрутизации пакетов из локальной сети](#)
24. [Добавление программ на нижнюю панель KDE](#)

На этой странице собраны рекомендации по эффективной работе в Calculate Linux.

Языковые настройки и параметры клавиатуры.

Переключение языка

В русской локализации Calculate Linux переключение языка выполняется клавишей *Caps Lock*, а фиксация верхнего регистра - сочетанием *Shift+CapsLock* (по аналогии с печатной машинкой).

Как поменять язык установленной системы?

Если во время загрузки LiveCD вы не выбрали свой язык, по умолчанию интерфейс будет англоязычным. Подробности см. в документации утилит Calculate.

Как отключить NumLock при входе в систему?

Выполните с правами рута в консоли:

```
rc-update del numlock
```

Чтобы вернуть включение NumLock при старте системы, выполните:

```
rc-update add numlock default
```

[Читать](#) подробнее о сценариях инициализации и о [rc-update](#) в частности.

Настройка звуковой карты

Если у вас не определилась звуковая карта или вы не слышите звука в динамиках, от имени пользователя *root* выполните команду

```
alsactl
```

с помощью которой определяются и настраиваются звуковые карты.

Создание пользователей

По умолчанию в [Calculate Linux Desktop](#) существует пользователь *guest*. Вы можете использовать эту учетную запись для знакомства с системой, а также для предоставления гостевого входа своим знакомым. Для создания новых учетных записей воспользуйтесь следующим [руководством](#).

Установка программ

При помощи нескольких [простых команд](#) вы можете обновлять или устанавливать новое программное обеспечение.

На компьютере хранится локальное дерево портежей со списком всех программ. Перед установкой или обновлением программ стоит обновить его с помощью команды

```
eix-sync
```

Ее нужно выполнять с правами пользователя *root*.

После этого можно установить новый пакет командой:

```
emerge имя_пакета
```

Узнать точное название пакета можно с помощью программы *eix*, выполнив:

```
eix часть_названия
```

Настройка маршрутизации пакетов из локальной сети

Если вам необходимо настроить Calculate Linux в качестве маршрутизатора для доступа в интернет локальных машин (192.168.0.0/24), выполните следующие команды:

```
sysctl -w net.ipv4.ip_forward=1
```

```
iptables -t nat -I POSTROUTING -s 192.168.0.0/24 -j MASQUERADE
```

Сохранить измененное значение `net.ipv4.ip_forward` можно в файле `/etc/sysctl.conf`.

Добавление программ на нижнюю панель KDE

1. Разблокируйте изменение виджетов, кликнув правой кнопкой на рабочем столе;
2. Кликните правой кнопкой на иконке меню слева вверху, выберите пункт "Редактировать меню";
3. Выберите и перетащите иконку приложения в нижнюю панель.

Руководство пользователя

Введение

Уважаемый пользователь! Благодарим за выбор операционной системы **Calculate Linux!**

Надеемся, что работа в новой системе не вызовет сложностей и доставит вам истинное удовольствие.

Рабочий стол

Рабочий стол Calculate Linux Desktop настроен с учетом комфортной работы пользователя. Каждый элемент рабочего стола (расположение панелей, порядок иконок, управляющие элементы) имеет свое особое место на экране, упрощая работу на компьютере. Большинство приложений предварительно настроены для экономии рабочего времени пользователя.

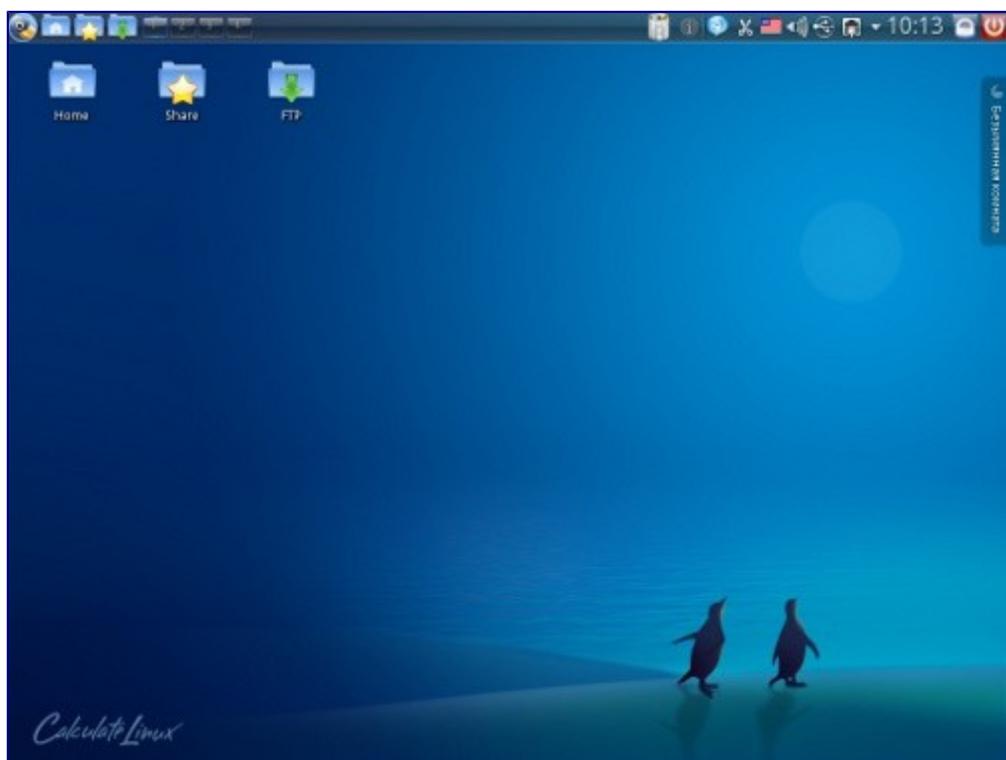


Рис. 1 Внешний вид рабочего стола Calculate Linux Desktop 11.0 KDE

Информационная панель

Верху рабочего стола находится *Информационная панель*.



Рис. 2 Информационная панель

Расположение элементов на панели слева направо:

1. Главное меню - аналог кнопки "Пуск" в ОС Windows.
2. Сетевые диски: *Home*, *Disks* и *FTP*.

3. *Переключатель рабочих столов*. Для быстрого переключения между рабочими столами.
4. *Панель задач*. Быстрый доступ к запущенным приложениями текущего рабочего стола.
5. *Корзина*. Содержит удаленные файлы и директории. Не забывайте иногда чистить корзину.
6. *Системный лоток*, отображает информацию некоторых приложений. В нем вы всегда найдете такие приложения как буфер обмена, организер, утилита для работы с клавиатурой, доступ к USB устройствам и т.д.
7. *Часы*. Если кликнуть на них мышкой, отобразится программа *Календарь*.
8. *Блокирование доступа и Завершение сеанса*. Покидая рабочее место, не забывайте блокировать доступ к компьютеру. Заблокировать доступ к компьютеру можно также по нажатию клавиши "Scroll Lock".

Панель приложений

Внизу по центру экрана находится *Панель приложений*.



Рис. 3 Панель приложений

Панель приложений включает все программы, необходимые в повседневной работе. Программы отсортированы по значению слева направо: Интернет-приложения, офис, дополнительные приложения, редактор, мультимедиа-приложения и утилиты.

Программное обеспечение

Почти всё программное обеспечение Calculate Linux Desktop имеет свободную лицензию, по которой вы можете получать доступ к исходным текстам программ, изменять код, распространять и даже продавать без каких-либо денежных отчислений правообладателям.

Основные программы

- Для навигации в интернете используется браузер *Chromium*.
- В качестве почтового клиента установлен *Kmail*.
- *Текстовый процессор* и *электронные таблицы* представлены офисным пакетом *LibreOffice*. В работе вы можете использовать также простой текстовый редактор *KWrite* (в версии CLD с рабочим окружением KDE).
- Программы, не имеющие аналогов в ОС Linux, могут работать на терминальном Windows-сервере, отображая информацию на экран вашего компьютера.

Дополнительные программы

Для доступа к остальным программам используйте значок *главного меню*, расположенный слева на верхней панели. Все приложения отсортированы по назначению, возле каждой программы находится пояснение.

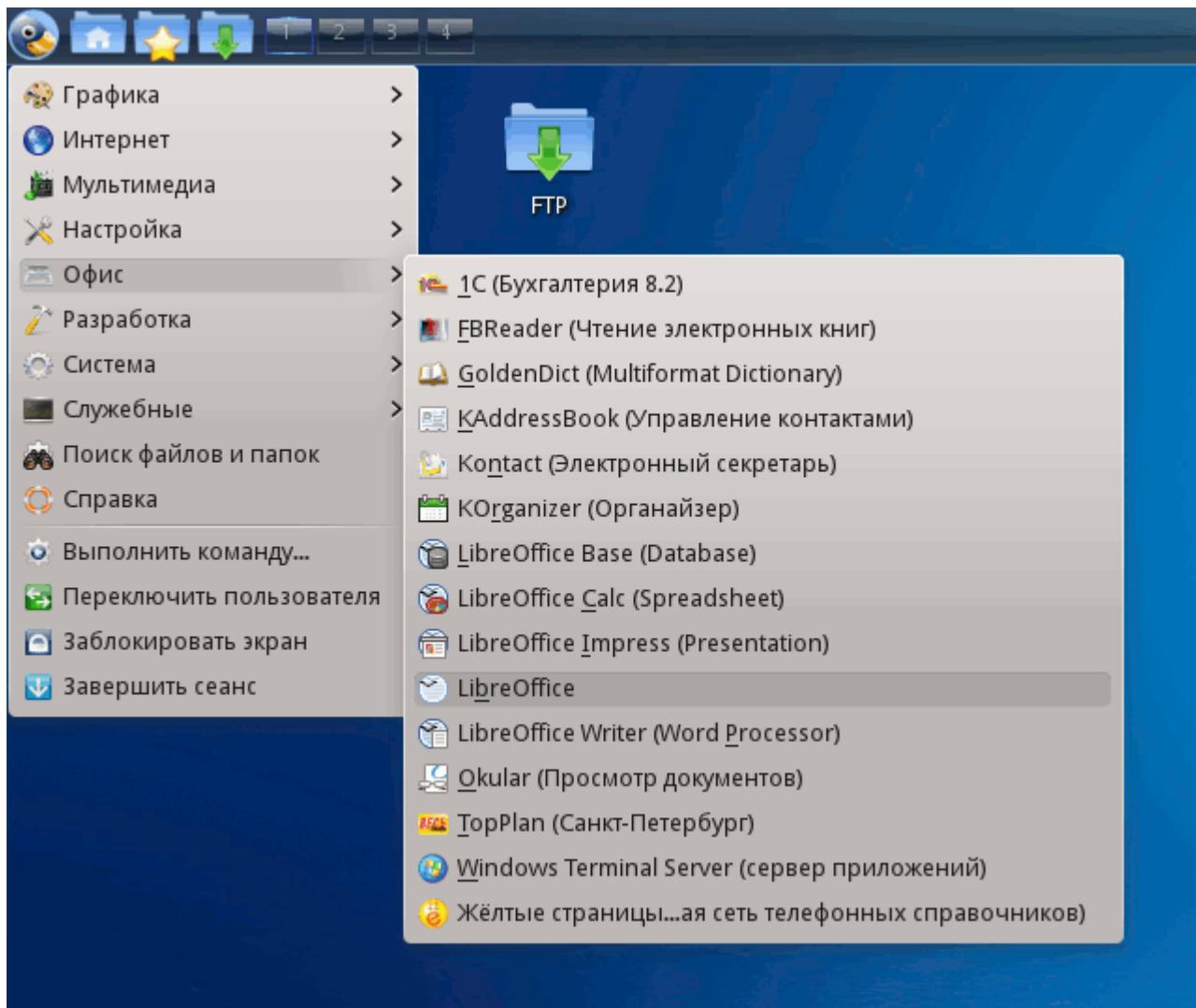


Рис. 4 Главное меню

Особенности работы

Рабочие столы

Используйте в своей работе несколько *рабочих столов*! На первом рабочем столе можно выполнять текущие задачи, на втором работать с документами, на третьем просматривать почту, переключаясь между ними мышкой (см. *Переключатель рабочих столов* верхней панели), либо при помощи комбинации клавиш *Ctrl+[F1-F4]*.

Доступ к файлам

Для доступа к файлам используйте иконки *Home*, *Disks* и *FTP* на рабочем столе или на верхней панели текущих задач. Папка *Home* открывает домашний диск пользователя, *Disks* - сетевые диски (общие ресурсы), *FTP* - файлы FTP сервера. Иконки *Disks* и *FTP* будут доступны в случае, если ваш компьютер введен в домен [Calculate Directory Server](#).

Раскладка клавиатуры

Раскладка клавиатуры аналогична раскладке в *ОС Windows*. Для переключения языка используйте *Caps Lock*. В то время, когда активна не английская раскладка, на клавиатуре будет гореть светодиод *Scroll Lock*. В системном лотке справа на информационной панели отображается выбранная раскладка.

Изменить регистр букв с нижнего на верхний и наоборот можно, нажав комбинацию клавиш *Shift+Caps Lock*.

CD/DVD, USB Flash

Для подключения носителя, вставьте CD-диск в дисковод или Flash-диск в USB порт компьютера. В системном лотке появится окно с доступными действиями.

Для завершения работы диск следует отмонтировать. Порядок завершения работы: кликните по иконке *Подключенные устройства*; выберите свое устройство из списка и кликните на значке отмонтирования. После этого устройство можно будет извлечь.

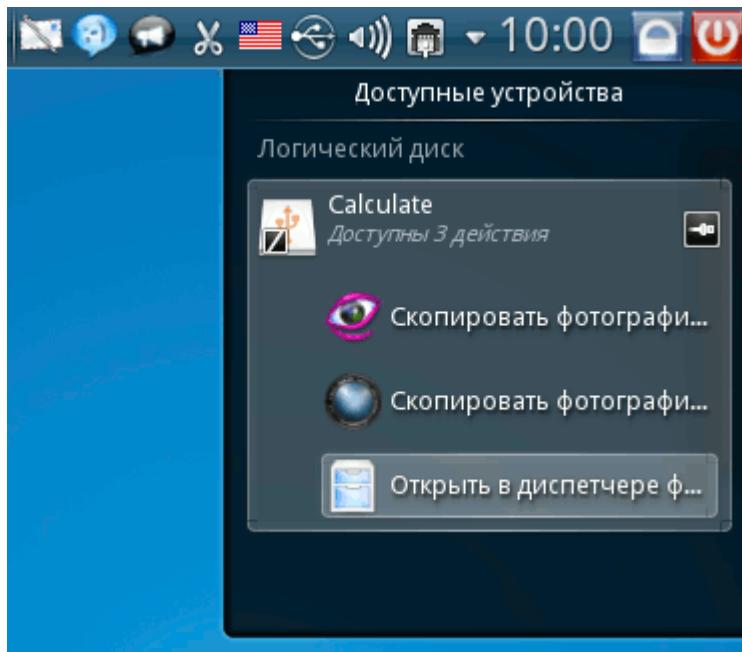


Рис. 5 Меню выбора действий над USB устройством

Буфер обмена

Calculate Linux Desktop имеет два буфера обмена. Первый работает аналогично ОС Windows. Копирование в буфер происходит путем нажатия сочетания клавиш *Ctrl+C* или *Ctrl+Insert*, извлечение - *Ctrl+V* или *Shift+Insert*.

Со вторым буфером обмена работа происходит исключительно мышкой. Во время выделения текста выделенный текст сразу же копируется в буфер обмена. Вставить этот текст можно так же легко, воспользовавшись третьей клавишей мыши (как правило, колесиком) - просто нажмите на колесико мыши в том месте, где нужно вставить текст. Если у вашей мыши нет колёсика или вы работаете с ноутбуком, третью клавишу заменяет одновременное нажатие двух кнопок.

Содержимое двух буферов обмена хранится раздельно. Можно получить доступ к истории буфера обмена, кликнув на ярлычке приложения буфера обмена в системном лотке или нажав сочетание клавиш *Ctrl+Alt+V*.

Создание учётных записей

Пользователи системы

После установки [Calculate Linux Desktop](#) в системе присутствует пользователь *guest*. Используйте учетную запись пользователя *guest* для знакомства с системой.

Пароль гостевого пользователя можно изменить в любой момент с помощью команды *passwd*. Удаленный доступ к компьютеру (по протоколу *ssh*) разрешен только для пользователя *root*.

Процесс смены паролей при установке системы на несколько компьютеров можно автоматизировать. Для этого установите систему на первый компьютер, измените пароли командой *passwd*, после чего сохраните файл */etc/shadow* в своем шаблоне.

Добавление пользователя

Linux неплохо справляется с контролем над действиями пользователя. Например, можно ограничить доступ к CD/DVD-приводу, звуковой карте, сканеру и даже компьютерным играм. Для наделения пользователя необходимыми полномочиями добавьте его в соответствующие системные группы.

Чтобы не подвергать систему риску, пользователю root запрещено работать в графическом окружении. Поэтому вам временно придется переключиться в консоль, нажав на клавиатуре Ctrl+Alt+F1. На приглашение ввести логин введите `root`, затем пароль. Далее выполните следующие команды:

```
/usr/sbin/useradd --create-home --groups  
users,wheel,audio,cdrom,video,cdrw,usb,plugdev,games,lp,scanner,iiusr  
<логин>  
/bin/passwd <логин>
```

В качестве логина вы можете использовать любое слово, состоящее из латинских букв и цифр.

При создании пользователя не следует создавать ему домашнюю директорию. В этом случае при первом входе в систему она создается автоматически с учетом настроенного шаблона пользователя. В приведенном примере создается пользователь с правами доступа к нескольким группам:

- `users` - доступ к менеджеру сети;
- `wheel` - возможность получить привилегии суперпользователя, используя команду `SU`;
- `audio` - предоставляет доступ к звуковой карте;
- `cdrom` - доступ к CD/DVD-приводу;
- `video` - доступ к TV-тюнеру;
- `cdrw` - запись на CD/DVD-диски;
- `usb` - доступ к mp3-плеерам, флеш-накопителям, доступ к USB в VirtualBox;
- `plugdev` - монтирование USB-устройств;
- `games` - доступ к играм;
- `scanner` - доступ к сканеру;
- `lp` - доступ к принтеру или сканеру, встроенному в МФУ;
- `iiusr` - доступ к модему.

Для возврата к графическому приглашению ввода пароля выйдите из сеанса, выполнив команду `exit` (либо нажав `Ctrl+D`), после чего нажмите `Alt+F7`.

Изменение прав доступа

Для того, чтобы добавить созданного пользователя в группу, воспользуйтесь командой `gpasswd`.

Пример добавления пользователя в группу `games`:

```
gpasswd -a <логин> games
```

Можно также напрямую редактировать файл `/etc/group`.

Перенос пользователей

При установке либо переустановке Calculate Linux Desktop из работающей Linux-системы все пользователи с их правами доступа будут перенесены в новую систему. Потребуется только повторно присвоить им пароли, используя команду `passwd`, по описанной выше схеме.

Шифрование домашних директорий

- [Шифрование домашних директорий](#)
- [Принцип работы](#)
- [Настройка профиля на шифрование](#)
- [Как восстановить данные](#)

Начиная с версии 13.6 в Calculate Linux появилась возможность использовать шифрованные домашние директории. Для организации защиты персональных данных используется eCryptfs - файловая система, работающая «поверх» любой другой обычной ФС и прозрачно шифрующая/десифрующая содержимое файлов. Криптографические метаданные eCryptfs хранят в заголовках каждого файла, таким образом, можно без проблем переносить любой файл между различными системами. Всё это реализовано на уровне ядра Linux, обеспечивая хороший уровень производительности по сравнению с FUSE-шифрованием.

Принцип работы

Домашняя папка пользователя с профилем, настроенным на шифрование, содержит только символические ссылки на директории с шифрованными данными. Шифрованные данные пользователей хранятся в каталоге `/home/.ecryptfs` и разделены по пользователям, при этом в папке `.Private` хранятся данные, в `.ecryptfs` - информация по подключению шифрования.

При входе в сеанс `/home/.ecryptfs/имя_пользователя/.Private` монтируется в домашнюю директорию. Для этого используется зашифрованный пользовательским паролем ключ: `/home/.ecryptfs/имя_пользователя/.ecryptfs/wrapped-passphrase`. Дальше происходят синхронизация профиля пользователя и настройка его при необходимости.

При выходе из сеанса домашняя директория отключается от `.Private`.

Данные доменных пользователей также могут быть настроены на шифрование, при этом нужно отметить, что в шифрованном виде эти данные будут находиться на локальной машине, на сервере они находятся в незашифрованном виде. Таким образом обеспечивается безопасность данных, оставленных в пользовательском профиле после выхода из сеанса.

Настройка профиля на шифрование

Для того, чтобы активировать данную функцию, систему нужно установить с включенным параметром "Шифровать пользовательские профили" (*Crypt user profiles*), либо включить переменную `main.cl_home_crypt_set` в уже установленной системе.

```
cl-core-variables --set main.cl_home_crypt_set=on
```

Шифрование будет настроено только для новых пользователей (тех, у которых нет домашней директории). Включить шифрование для существующего профиля можно при помощи команды `ecryptfs-migrate-home`.

```
ecryptfs-migrate-home -u <имя_пользователя>
```

Команда запросит пароль пользователя.

Для доменных пользователей будет достаточно удалить локальный профиль, когда он не в сеансе, и после этого войти в сеанс.

Стоит отметить, что на настроенный пользовательский профиль значение переменной `cl_home_crypt_set` не влияет, то есть если профиль был настроен на шифрование, а система была установлена без шифрования, профиль будет продолжать работать с шифрованием, и наоборот.

Как восстановить данные

Если в `/home` остался только каталог `.ecryptfs`, содержащий шифрованные профили, то достаточно создать таких же пользователей с теми же паролями, и при входе в сеанс домашний каталог пользователя будет использоваться существующий шифрованный профиль.

Установка и удаление программ

- [Установка и удаление программ](#)
- [Обновление дерева портежей](#)
- [Поиск программ](#)

- [Установка и удаление](#)
- [Размаскировка программ](#)
- [1. Проверим доступные версии](#)
- [2. Размаскируем устанавливаемые пакеты с зависимостями](#)
- [3. Установим программу](#)

Обновление дерева портежей

Перед установкой программ обновите локальный репозиторий пакетов. Обновлять репозиторий следует не чаще 1 раза в день.

Для обновления достаточно выполнить с правами пользователя root команду:

```
cl-update --sync-only
```

Программа обновит дерево портежей, оверлей Calculate, а затем синхронизирует свой локальный кэш, используемый при поиске программ.

Поиск программ

В программу emerge включен инструмент поиска программ, однако вы можете воспользоваться более быстрым инструментом - программой eix.

Пример:

```
eix freeciv  
eix -S game
```

В первом случае поиск производится по названию пакета, во втором - по описанию.

Установка и удаление

Установка и удаление программ производится при помощи программы [emerge](#). При установке новой программы сначала определяется необходимость в установке дополнительных пакетов (зависимостей), а затем скачиваются и компилируются исходные тексты. Помните, что для установки программ вам потребуется наличие интернета и некоторое время для компиляции. Программы, требующие значительного времени для компиляции (например, LibreOffice), распространяются в виде готовых к установке бинарных пакетов; их можно опознать по суффиксу "-bin".

Пример установки игры "Цивилизация" и бинарного LibreOffice:

```
emerge -bk games-strategy/freeciv  
emerge libreoffice-bin
```

Параметры -bk создают локальный архив скомпилированного пакета, а при его наличии программа устанавливается из этого архива, минуя стадию компиляции. Инструкцию по работе с программой emerge можно прочесть [здесь](#).

Пример удаления игры "Цивилизация":

```
emerge -C games-strategy/freeciv
```

Переменные DISTDIR и PKGDIR указывают путь к локальным папкам, в которых сохраняются исходные тексты программ и откомпилированные пакеты. Посмотреть значения этих (и многих других) переменных, используемых emerge, можно с помощью команды

```
emerge --info
```

Размаскировка программ

Как правило, портежи содержат несколько версий программы, часть из которых отмечены как нестабильные. Нестабильная версия вовсе не означает, что она нестабильно работает - просто на данный момент эта версия не прошла достаточного тестирования. Вы можете установить как стабильную, так и нестабильную версию программы.

Для установки нестабильных программ их нужно предварительно размаскировать. Для примера установим нестабильный на данный момент Firefox 22.0.

1. Проверим доступные версии

```
eix firefox
www-client/firefox
    Available versions: *10.0.11 17.0.5 17.0.6 17.0.7 ~21.0 ~22.0
...
```

2. Размаскируем устанавливаемые пакеты с зависимостями

Удостоверьтесь, что директория /etc/portage/package.keywords/ не пустая, иначе выполните:
touch /etc/portage/package.{keywords,unmask,use}/custom

Выполните размаскировку:

```
emerge --autounmask-write =www-client/firefox-22.0
```

Обновите настройки:

```
dispatch-conf
```

Нажмите "u" для подтверждения внесенных изменений.

3. Установим программу

```
emerge firefox
```

- [Руководство по обновлению системы](#)
- [Обновление из пакетов](#)
- [1. Обновление оверлея и портежей](#)
- [2. Обновление программ](#)
- [3. Обновление файлов настроек](#)
- [Обновление из ISO образа](#)
- [1. Обновите установщик](#)
- [2. Загрузите ISO образ последней Stage-сборки](#)
- [3. Установите новую версию системы](#)

Руководство по обновлению системы

Для управления пакетами Calculate Linux использует [Portage](#). Вы можете обновить систему двумя способами:

- Обновление системы из пакетов;
- Обновление из ISO образа.

Обновление из пакетов

Порядок обновления:

1. Обновление оверлея и портежей

Репозиторий бинарных пакетов постоянно обновляется, поэтому важно перед установкой или обновлением пакетов иметь свежую версию портежей и оверлея `calculate`.

Выполнить обновление можно одной командой:

```
cl-update --sync-only
```

После запуска последовательно будут выполнены следующие действия:

- обновятся репозитории;
- обновятся портежи;
- обновлены шаблоны;
- применены ревизии.

Если у вас подключены сторонние оверлеи, то нужно также указать опцию "`--update-other`" ("`-o`"):

```
cl-update --sync-only --update-other
```

2. Обновление программ

Каждый дистрибутив Calculate Linux имеет собственный репозиторий бинарных пакетов, оптимизированными под конкретный дистрибутив. По умолчанию обновление производится именно из бинарных пакетов. Изменить способ обновления по умолчанию на обновление из исходных кодов можно добавив в файл `/etc/portage/make.conf/custom` следующую строку:

```
FEATURES="-getbinpkg"
```

Дистрибутивы Calculate Linux имеют непрерывный цикл обновлений, отличаются составом пакетов, USE-флагами и масками. Внутренние настройки системы находятся в профиле дистрибутива. Список доступных профилей для вашей архитектуры можно посмотреть, выполнив:

```
cl-update-profile list
```

Изменить профиль можно также командой `cl-update-profile`, указав имя профиля, например:

```
cl-update-profile CLDX
```

Вы можете использовать профиль из стороннего репозитория, для этого воспользуйтесь параметром "`--url`".

В Calculate Linux 14 появилась утилита для обновления системы `cl-update` - документацию по ней вы можете прочитать на следующей [странице](#).

Если вам нужно обновить всю систему целиком, включая обновление оверлеев и портежей, то просто используйте команду:

```
cl-update
```

Порядок обновления в общем случае следующий:

1. Синхронизация репозиториев дистрибутива
2. Если репозитории были обновлены, то выполняются действия `eugenecache` и `eix-update`
3. Обновление ревизии и обновление мира
4. Обновление системы
5. Обновление Python/Perl с пересборкой поврежденных пакетов при необходимости
6. Удаление ненужных пакетов
7. Пересборка модулей ядра при необходимости
8. Пересборка прочих поврежденных пакетов при необходимости
9. Пересборка пакетов для Xorg-сервера, если в этом есть необходимость
10. Выполнение `dispatch-conf`

3. Обновление файлов настроек

По умолчанию, во время обновления системы конфигурационные файлы программ не переписываются, если вы вносили в них изменения. При обнаружении новых изменений вам будет предложено несколько действий: "PageUp"/"PageDown" - перемещаться по файлу, "u" - заменить существующий файл новым, "z" - удалить новый конфигурационный файл, "q" - прервать работу.

Вы также можете использовать автозамену настроек конфигурационных файлов программ при установке значения переменной `cl_autoupdate_set` в файле `/etc/calculate/calculate.env`:

```
[main]
cl_autoupdate_set = on
```

В этом случае внимательно относитесь к производимым модификациям файлов, используя для этого шаблоны.

Обновление из ISO образа

Обновить систему можно путем установки нового образа в свободный системный раздел. Основные настройки, такие как учетные записи пользователей, настройки сети, точки монтирования, разрешение экрана и прочие, будут перенесены, дополнительные настройки будут выполнены при помощи шаблонов также на этапе установки.

Если у вас установлен [Calculate Directory Server](#), удостоверьтесь, что директория `/var/calculate` монтируется с [отдельного раздела](#) жесткого диска. Если это не так, перенесите свои данные и добавьте точку монтирования в `/etc/fstab`.

Порядок обновления:

Откройте консоль с правами пользователя `root` и выполните следующие действия:

1. Обновите установщик

Для корректного обновления всегда используйте последнюю доступную версию пакета `calculate-install`. Чтобы обновить программу, выполните:

```
cl-update -s && emerge calculate-utils
```

2. Загрузите ISO образ последней Stage-сборки

На [HTTP](#) и [FTP](#) зеркалах в директории `stages` доступны еженедельные сборки дистрибутивов.

Скачайте последний доступный образ:

```
mkdir -p /var/calculate/remote/linux
cd /var/calculate/remote/linux
wget http://mirror.yandex.ru/calculate/CLD/stages/i686/cld-*****-i686.iso
```

Подставьте [правильный путь](#) к файлу с образом вашего дистрибутива нужной архитектуры.

3. Установите новую версию системы

```
cl-install
```

Если вы обновляете Calculate Directory Server, сохраните копию настроек сервисов и базы `LDAP`, выполнив:

```
cl-backup
```

Перезагрузите компьютер. Для восстановления LDAP-базы и настроек сервера выполните:

К основным преимуществам такого способа обновления можно отнести следующие:

- надежность - вы всегда сможете загрузиться в предыдущую систему, если новая по какой-либо причине будет работать нестабильно;
- скорость обновления - потребуется всего 5-7 минут на полное обновление системы.

Состав программного обеспечения ISO образа можно предварительно модифицировать, воспользовавшись руководством по [интерактивной сборке системы](#).

Сценарии инициализации

- [Сценарии инициализации](#)
- [Уровни запуска](#)
- [Процесс загрузки системы](#)
- [Сценарии инициализации](#)
- [Как работает init](#)
- [Что такое уровень запуска?](#)
- [Работа со сценариями инициализации](#)
- [Использование rc-update](#)
- [Что такое rc-update?](#)
- [Добавление и удаление служб](#)
- [Настройка служб](#)
- [Почему нужна дополнительная настройка?](#)
- [Каталог /etc/conf.d](#)
- [Написание сценариев инициализации](#)
- [Мне тоже придется?..](#)
- [Структура](#)
- [Зависимости](#)
- [Порядок запуска](#)
- [Стандартные функции](#)
- [Добавление дополнительных параметров](#)
- [Переменные для настройки служб](#)
- [Кто от этого выигрывает?](#)
- [Использование программного уровня \(softlevel\)](#)
- [Использование загрузочного уровня \(bootlevel\)](#)

Уровни запуска

Процесс загрузки системы

При загрузке вашей системы по экрану пробегает много текста. Если присмотреться, заметно, что этот текст не меняется от загрузки к загрузке. Последовательность всех этих действий называется *последовательностью загрузки* и в той или иной степени постоянна.

Во-первых, загрузчик размещает в памяти образ ядра, который вы указали в файле его конфигурации. После этого ядро запускается. Когда ядро загружено и запущено, оно инициализирует относящиеся к ядру структуры и задания, и запускает процесс init.

Этот процесс удостоверяется, что все файловые системы (определенные в `/etc/fstab`) смонтированы и готовы к использованию. Затем он выполняет несколько сценариев, находящихся в каталоге `/etc/init.d`, которые запускают службы, необходимые для нормального запуска системы.

И, наконец, когда все сценарии выполнены, `init` подключает терминалы (чаще всего просто виртуальные консоли, которые видны при нажатии `ALT+F1`, `ALT+F2` и т.д.), прикрепляя к каждой консоли специальный процесс под названием `agetty`. Этот процесс впоследствии обеспечивает возможность входа в систему с помощью `login`.

Сценарии инициализации

Сейчас процесс `init` запускает сценарии из каталога `/etc/init.d` не в случайном порядке. Более того, запускаются не все сценарии из `/etc/init.d`, а только те, которые предписано выполнять. Решение о запуске сценария принимается в результате просмотра каталога `/etc/runlevels`.

Во-первых, `init` запускает все сценарии из `/etc/init.d`, на которые есть символьные ссылки из `/etc/runlevels/boot`. Обычно сценарии запускаются в алфавитном порядке, но в некоторых сценариях имеется информация о зависимостях от других сценариев, указывающая системе на необходимость их предварительного запуска.

Когда все сценарии, указанные в `/etc/runlevels/boot`, будут выполнены, `init` переходит к запуску сценариев, на которые есть символьные ссылки из `/etc/runlevels/default`. И снова запуск происходит в алфавитном порядке, пока в сценарии не встретится информация о зависимостях; тогда порядок изменяется для обеспечения правильного порядка запуска.

Как работает `init`

Конечно, `init` не принимает решений сам по себе. Ему необходим конфигурационный файл, где описаны необходимые действия. Этот файл --- `/etc/inittab`.

Если вы запомнили последовательность загрузки, описанную чуть ранее, вы вспомните, что первое действие `init` --- это монтирование всех файловых систем. Это определяется в строке `/etc/inittab`, приведенной ниже:

```
si::sysinit:/sbin/rc sysinit
```

Этой строкой процессу `init` предписывается выполнить `/sbin/rc sysinit` для инициализации системы. Самой инициализацией занимается сценарий `/sbin/rc`, так что можно сказать, что `init` делает не слишком много --- он просто делегирует задачу по инициализации системы другому процессу.

Во-вторых, `init` выполняет все сценарии, на которые есть символьные ссылки из `/etc/runlevels/boot`. Это определяется следующей строкой:

```
rc::bootwait:/sbin/rc boot
```

И снова все необходимые действия выполняются сценарием `rc`. Заметьте, что параметр, переданный `rc` (`boot`), совпадает с названием используемого подкаталога в `/etc/runlevels`.

Теперь `init` проверяет свой конфигурационный файл, чтобы определить, какой уровень запуска использовать. Для этого из `/etc/inittab` считывается строка:

```
id:3:initdefault:
```

В приведенном примере (который подходит для подавляющего большинства пользователей Calculate) номер уровня запуска --- 3. Пользуясь этой информацией, `init` проверяет, что нужно выполнить для запуска уровня запуска 3. Пример уровней запуска:

```
10:0:wait:/sbin/rc shutdown
11:S1:wait:/sbin/rc single
12:2:wait:/sbin/rc nonetwork
13:3:wait:/sbin/rc default
14:4:wait:/sbin/rc default
15:5:wait:/sbin/rc default
16:6:wait:/sbin/rc reboot
```

В строке, определяющей уровень 3, для запуска служб снова используется сценарий rc (на этот раз с аргументом *default*). Опять-таки, обратите внимание, что аргумент, передаваемый сценарию rc, совпадает с названием подкаталога из /etc/runlevels.

По окончании работы rc, init принимает решение о том, какие виртуальные консоли включить и какие команды выполнить в каждой из них. Пример определения виртуальных консолей:

```
c1:12345:respawn:/sbin/agetty 38400 tty1 linux
c2:12345:respawn:/sbin/agetty 38400 tty2 linux
c3:12345:respawn:/sbin/agetty 38400 tty3 linux
c4:12345:respawn:/sbin/agetty 38400 tty4 linux
c5:12345:respawn:/sbin/agetty 38400 tty5 linux
c6:12345:respawn:/sbin/agetty 38400 tty6 linux
```

Что такое уровень запуска?

Как вы заметили, init применяет нумерацию для определения уровня запуска, который надо использовать. Уровень запуска --- это то состояние, в котором запускается ваша система; он содержит набор сценариев (сценариев уровня запуска или *сценариев инициализации [initscript]*), которые следует выполнять при входе и выходе из определенного уровня запуска.

В Calculate определено семь уровней запуска: три служебных и четыре определяемых пользователем. Служебные называются *sysinit*, *shutdown* и *reboot*. Действия, совершаемые ими, в точности соответствуют их названиям: инициализация системы, выключение системы и ее перезагрузка.

Определяемые пользователем уровни --- это те, которым соответствуют подкаталоги в /etc/runlevels: *boot*, *default*, *nonetwork* и *single*. Уровень *boot* запускает все службы, необходимые системе и используемые всеми остальными уровнями. Остальные уровни отличаются друг от друга запускаемыми службами: *default* используется для повседневной работы, *nonetwork* --- для тех случаев, когда не требуется сеть, а *single* --- при необходимости восстановления системы.

Работа со сценариями инициализации

Сценарии, запускаемые процессом rc, называются *сценариями инициализации*. Каждый сценарий из /etc/init.d может запускаться с аргументами *start*, *stop*, *restart*, *pause*, *zap*, *status*, *ineed*, *iuse*, *needsme*, *usesme* и *broken*.

Для запуска, остановки или перезапуска службы (и всех, зависящих от нее) следует использовать *start*, *stop* и *restart*. Пример запуска postfix:

```
/etc/init.d/postfix start
```

Примечание: Останавливаются или перезапускаются только те службы, которым *необходима* данная служба. Остальные зависимые службы (те, которые *используют* службу, но не нуждаются в ней) эта операция не затрагивает.

Если вы хотите остановить службу, но оставить зависимые от нее работающими, можно использовать аргумент *pause*. Пример:

```
/etc/init.d/postfix pause
```

Чтобы узнать текущее состояние службы (запущена, остановлена, приостановлена и т.д.), можно использовать аргумент *status*. Пример:

```
/etc/init.d/postfix status
```

Если указано, что служба работает, но вы знаете, что это не так, можно сбросить состояние на *stopped* (остановлена), используя аргумент *zap*. Пример сброса информации о состоянии postfix:

```
/etc/init.d/postfix zap
```

Для того, чтобы выяснить зависимости службы, можно использовать аргументы `iuse` или `ineed`. С помощью `ineed` вы увидите те службы, которые действительно необходимы для правильного функционирования интересующей вас службы. С другой стороны, `iuse` покажет те службы, которые могут использоваться нашей службой, но не обязательны для ее работы. Пример запроса списка всех необходимых служб, от которых зависит Postfix:

```
/etc/init.d/postfix needsme
```

Наконец, можно просмотреть список служб, требующихся для данной, но отсутствующих в системе. Пример запроса списка служб, необходимых Postfix, но отсутствующих:

```
/etc/init.d/postfix broken
```

Использование rc-update

Что такое rc-update?

Система инициализации Calculate использует дерево зависимостей для определения служб, которые запускаются в первую очередь. Т. к. это очень утомительное занятие, и мы не хотели, чтобы пользователь занимался этим вручную, были разработаны инструменты, упрощающие управление уровнями запуска и сценариями инициализации.

Используя `rc-update`, можно включать и исключать сценарии инициализации из уровней запуска. Из `rc-update` автоматически запускается сценарий `depscan.sh`, который перестраивает дерево зависимостей.

Добавление и удаление служб

В процессе установки Calculate вы могли добавлять сценарии инициализации в уровень запуска "default". В тот момент вы, возможно, не имели понятия, что такое "default" и зачем он нужен, но теперь вы это знаете. Сценарию `rc-update` требуется второй аргумент, определяющий действие: `add` (добавить), `del` (удалить) или `show` (показать).

Для того, чтобы добавить или удалить сценарий, просто введите `rc-update` с аргументом `add` или `del`, затем название сценария и уровня запуска. Пример удаления Postfix из уровня запуска default:

```
rc-update del postfix default
```

По команде `rc-update show` выводится список всех доступных сценариев с указанием соответствующих уровней запуска. Пример получения информации о сценариях инициализации:

```
rc-update show
```

Настройка служб

Почему нужна дополнительная настройка?

Сценарии инициализации могут быть весьма сложны. Поэтому нежелательно допускать непосредственное редактирование сценария пользователями, т.к. это может привнести в систему множество ошибок. Но, с другой стороны, необходимо правильно настроить службу. Например, может понадобиться передать службе дополнительные параметры.

Вторая причина, по которой настройки хранятся отдельно от самого сценария --- это возможность обновления сценария без опасения, что все ваши настройки будут утеряны.

Каталог /etc/conf.d

В Calculate предусмотрен очень простой способ настройки служб: для каждого сценария, предполагающего настройку, в каталоге `/etc/conf.d` есть конфигурационный файл. Например, у сценария, запускающего apache2 (под названием `/etc/init.d/apache2`), есть конфигурационный файл `/etc/conf.d/apache2`, где могут храниться нужные вам параметры, передаваемые серверу Apache 2 при запуске. Пример переменной, определенной в `/etc/conf.d/apache2`:

Такие файлы настроек содержат одни переменные (наподобие [/etc/make.conf](#)), облегчая настройку служб. Это также позволяет нам давать больше информации о переменных (в комментариях).

Написание сценариев инициализации

Мне тоже придется?..

Нет, написание сценариев инициализации обычно не требуется, т.к. Calculate содержит готовые сценарии для всех поддерживаемых служб. Однако, вы можете установить какую-либо службу, не используя систему Portage; в таком случае, вероятно, вам придется создавать сценарий инициализации самостоятельно.

Структура

Основная структура сценария инициализации показана ниже.

```
#!/sbin/runscript

depend() {
    (информация о зависимостях)
}

start() {
    (команды, необходимые для запуска службы)
}

stop() {
    (команды, необходимые для остановки службы)
}

restart() {
    (команды, необходимые для перезапуска службы)
}
```

В любом сценарии должна быть определена функция start(). Все остальные разделы необязательны.

Зависимости

Можно определять два типа зависимостей: use (использую) и need (нуждаюсь). Как упоминалось ранее, need-зависимость более строга, чем use-зависимость. Вслед за типом зависимости указывается название службы, от которой существует зависимость, или ссылка на *виртуальную* (virtual) зависимость.

Виртуальная зависимость --- это зависимость от функций, предоставляемых службой, но не какой-то единственной службой. Сценарий может зависеть от службы системного журнала, но таких достаточно много (metalogd, syslog-ng, sysklogd и т.п.). Поскольку нельзя нуждаться в каждой из них (ни в одной корректно работающей системе они не запущены все сразу), мы обеспечили *предоставление* виртуальной зависимости всеми этими службами.

Давайте взглянем на информацию о зависимостях postfix:

```
depend() {
    need net
    use logger dns
    provide mta
}
```

Как можно увидеть, postfix:

- требует сеть (net): виртуальная зависимость, удовлетворяемая, например, `/etc/init.d/net.eth0`
- использует журнал (logger): виртуальная зависимость, удовлетворяемая, например, `/etc/init.d/syslog-ng`
- использует службу имен (dns): виртуальная зависимость, удовлетворяемая, например, `/etc/init.d/named`
- предоставляет почтовый агент (mta): виртуальная зависимость, общая для всех программ --- почтовых серверов

Порядок запуска

Иногда вам нужна не сама служба, а запуск вашей службы до (или *после*) другой службы, *если* та присутствует в системе (обратите внимание на условие: это уже не зависимость) и запускается на том же уровне запуска (отметьте условие: это относится только к службам из одинакового уровня запуска). Такую очередьность можно указать, используя значения *before* (до) или *after* (после).

Например, рассмотрим значения для службы Portmap. Пример функции `depend()` службы Portmap:

```
depend() {
    need net
    before inetd
    before xinetd
}
```

Также можно использовать знак "*", чтобы охватить все службы данного уровня запуска, хотя это не рекомендуется. Пример запуска сценария первым на уровне запуска:

```
depend() {
    before *
}
```

Стандартные функции

Следом за разделом `depend()` вам потребуется определить функцию `start()`. В ней содержатся все команды, необходимые для запуска вашей службы. Рекомендуется применять функции `ebegin` и `eend` для сообщений пользователю о том, что происходит. Пример функции `start()`:

```
start() {
    ebegin "Запуск - моя_служба"
    start-stop-daemon --start --quiet --exec /path/to/my_service
    eend $?
}
```

Если вам нужны дополнительные примеры функции `start()`, пожалуйста, прочтайте исходные коды сценариев инициализации, находящихся в каталоге `/etc/init.d`. Что касается команды `start-stop-daemon`, то на случай, если вам нужны дополнительные сведения, есть превосходная страница справки. Пример вызова страницы справки по `start-stop-daemon`:

```
man start-stop-daemon
```

Другими функциями, которые можно определить --- `stop()` и `restart()`. От вас не требуется определение этих функций! Система инициализации, применяемая нами, достаточно развита и в состоянии самостоятельно заполнить эти функции, если вы используете `start-stop-daemon`.

Синтаксис сценариев инициализации, применяемых в Calculate, основан на оболочке Борна (Bourne Again Shell --- bash), поэтому вы можете свободно использовать внутри своих сценариев bash-совместимые конструкции.

Добавление дополнительных параметров

Если вы хотите ввести в сценарий дополнительные параметры, кроме упоминавшихся, нужно добавить к переменной opts название параметра и создать функцию с названием, соответствующим параметру. Например, для поддержки параметра restartdelay. Пример создания дополнительной функции restartdelay:

```
opts="${opts} restartdelay"

restartdelay() {
    stop
    sleep 3      # пауза в 3 секунды перед повторным запуском
    start
}
```

Переменные для настройки служб

Для поддержки конфигурационного файла в каталоге `/etc/conf.d` ничего дополнительно делать не нужно: при запуске вашего сценария инициализации автоматически включаются следующие файлы (т.е., переменные из них становятся доступны):

- `/etc/conf.d/<ваш сценарий инициализации>`
- `/etc/conf.d/basic`
- `/etc/rc.conf`

Если ваш инициализационный сценарий предоставляет виртуальную зависимость (например, `net`), то также включается файл, соответствующий этой зависимости (например, `/etc/conf.d/net`).

4.e. Изменение поведения уровней запуска

Кто от этого выиграет?

Большинству пользователей ноутбуков знакома ситуация: дома вам нужен запуск `net.eth0`, а в дороге, наоборот, запуск `net.eth0` не нужен (так как сеть недоступна). В Calculate можно изменять поведение уровней запуска по своему усмотрению.

Например вы можете создать второй загружаемый уровень запуска «по умолчанию», в котором будут другие сценарии. Затем при загрузке вы сможете выбрать, какой из уровней по умолчанию следует использовать.

Использование программного уровня (softlevel)

Прежде всего, создайте каталог для своего второго уровня запуска «по умолчанию». Например, создадим уровень запуска `offline`. Пример создания каталога уровня запуска:

```
mkdir /etc/runlevels/offline
```

Добавьте необходимые сценарии инициализации в только что созданный уровень запуска. Например, чтобы получить точную копию уровня `default`, за исключением `net.eth0`:

```
(копирование всех служб с уровня default в уровень offline)
# cd /etc/runlevels/default
# for service in *; do rc-update add $service offline; done
(удаление ненужных сценариев с уровня offline)
# rc-update del net.eth0 offline
(просмотр сценариев, запускаемых на уровне offline)
# rc-update show offline
(часть выведенного списка)
        acpid | offline
        domainname | offline
        local | offline
```

Теперь необходимо отредактировать конфигурацию загрузчика, добавив запись об уровне offline. Например, в файле /boot/grub/grub.conf. Пример:

```
title Автономное использование Calculate Linux
root (hd0, 0)
kernel /boot/vmlinuz-5bf7e746 root=/dev/hda3 softlevel=offline
```

Вуаля, все готово. Теперь, если при загрузке вы выберете вновь созданную запись, то вместо default будет использоваться уровень offline.

Использование загрузочного уровня (bootlevel)

Использование загрузочного уровня полностью аналогично использованию *программного уровня*. Единственная разница состоит в том, что вы определяете второй уровень "boot" вместо "default".

Сборка ядра со своей конфигурацией

- [Сборка ядра со своей конфигурацией](#)
- [Введение](#)
- [cl-kernel](#)
- [Сборка ядра](#)
- [Обновление ядра](#)
- [Настройка calculate-sources](#)
- [Оптимизация ядра](#)
- [Использование патчей](#)
- [Для разработчиков](#)

Введение

Calculate Linux работает на ядре Linux с длительным сроком поддержки (longterm). Большая часть драйверов вынесена в модули, что позволяет ядру оставаться компактным в размере без потери функционала. Для серверов и десктопов используются различные настройки и патчи. В отличие от других ядер из портфолио, пакет sys-kernel/calculate-sources, используемый по умолчанию в дистрибутивах Calculate Linux, компилируется и устанавливается в систему, как и большинство других пакетов, высвобождая место путём удаления за собой большей части исходного кода.

Зачем вам может понадобиться изменять настройки ядра? Оптимизируя ядро, вы можете добиться прироста производительности, поддержки своего железа, высвобождения памяти, снижения энергопотребления, а так же ускорения загрузки системы. Помимо прочего, изучение ядра даёт неплохие знания в понимании устройства работы системы.

Экспериментировать с ядром можно и нужно. Эта статья поможет вам научиться изменять настройки ядра, устанавливать и настраивать различные модификации ядер, использовать свои патчи.

cl-kernel

Для сборки ядра служит скрипт cl-kernel, входящий в состав пакета sys-apps/calculate-toolkit. Программа написана на Bash и прозрачно интегрирована с системой шаблонов утилит Calculate.

Особенности

1. Поддержка сборки различных ядер: sys-kernel/calculate-sources, sys-kernel/gentoo-sources, sys-kernel/vanilla-sources и др.
2. Поддержка создания ядра, как с использованием initramfs, так и без него.
3. Создание шаблона настроек ядра со всеми внесёнными изменениями.
4. Импорт готовых настроек ядра в шаблон.

5. Прозрачная миграция настроек между версиями ядер.
6. Интеграция с утилитами Calculate для использования шаблонов настроек во время установки ядра `calculate-sources`.
7. Создание резервных копий настроек.
8. Локализация на русский и французский языки.

Прежде чем приступить к дальнейшим действиям, убедитесь, что у вас достаточно свободного места на диске. Исходный код ядра распаковывается в директорию `/usr/src`. Посмотреть свободное место можно, выполнив:

```
df -h
```

Обязательно позаботьтесь о наличии резервной копии ядра, с которого всегда можно загрузить систему. Для этого эксперименты удобно проводить с альтернативными пакетами ядер, либо с ядром `calculate-sources` не установленной версии.

Сборка ядра

Выберите любое из доступных в портежах ядро. Весь список с описаниями можно посмотреть, выполнив:

```
eix -c sys-kernel/*sources
```

Для примера остановим свой выбор на "ванильном" ядре - оригинальной версии, поддерживаемой Линусом Торвальдсом.

Ядро в портежах отмечено маской, поэтому понадобится сперва её снять:

```
echo sys-kernel/vanilla-sources ~amd64 ~x86 >>
/etc/portage/package.keywords/custom
USE="symlink" emerge sys-kernel/vanilla-sources
```

USE-флаг "symlink" следует устанавливать, если вы используете проприетарные пакеты, такие как `nvidia-drivers`, `ati-drivers`, `virtualbox-bin` или `broadcom-sta`. В этом случае после установки ядра следует собрать их модули, выполнив:

```
emerge @module-rebuild
```

Проверьте, что ваше ядро стало доступно:

```
cl-kernel --kver list
* 3.19.0 *
* 3.18.7-calculate
```

Обратите внимание на список. В отличие от `calculate-sources` (и других ядер), ванильное ядро не содержит слова "vanilla". Красная звёздочка слева версии ядра означает, что установленное ядро не содержит полной версии исходного кода. Звёздочка справа отмечает используемое по умолчанию ядро. Оно определяется по символьской ссылке `/usr/src/linux`.

Для первого запуска вы можете сконвертировать настройки из текущей версии ядра, для этого выполните:

```
cl-kernel --kver 3.19.0 --convert
```

Здесь важно понять особенность работы `cl-kernel` и её отличие от более ранней версии этой программы.

1. Скрипт `cl-kernel` работает с конфигурационным файлом ядра, полученным из шаблона.
2. По завершении работы программа проанализирует изменения, выполненные пользователем, и создаст новый пользовательский шаблон.

3. Для ядер, отличных от `calculate-sources`, нет шаблонов настроек, поэтому готовый шаблон будет содержать отличия от настроек ядра по умолчанию.
4. С опцией "`--convert`" программа возьмёт за основу настройки текущего ядра (из `/boot` или `/proc`), если в директории с исходным кодом ядра нет файла `".config"`.

Во время выполнения скрипта будет вызвана настройка ядра (вызов `make menuconfig`), сборка и установка. Если не отключена опция `CONFIG_BLK_DEV_INITRD`, будет создан `initramfs`.

После завершения не забудьте обновить необходимые модули, выполнив:

```
emerge @module-rebuild
```

Теперь можно перезагрузиться, чтобы проверить работу нового ядра! Во время загрузки обратите внимание, что загружается новое ядро. Если вы ничего не изменили в окне настроек, проблем с загрузкой возникнуть не должно.

Обновление ядра

Посмотрите на первую строку шаблона, который `cl-kernel` создал из исходного файла настроек ядра:

```
head -n 1 /var/calculate/templates/kernel/10-vanilla-x86_64-3.19
# Calculate format=kernel name=.config
os_install_arch_machine==x86_64&&merge(sys-kernel/vanilla-sources)>=3.19
```

Первая строка - это заголовок шаблона. В ней описан формат шаблона, имя настраиваемого файла, выполняется проверка на архитектуру системы, имя и версию ядра.

Из шаблона видно, что он будет работать для всех ядер версии 3.19 и выше.

Для установки 3.19.1 ядра, после установки пакета достаточно будет выполнить:

```
cl-kernel --kver=3.19.1
```

Параметр '`--kver`' можно опустить, если ядро выбрано по умолчанию, когда символьическая ссылка `/usr/src/linux` указывает на него. Так будет, если при установке пакета с исходным кодом ядра, вы указали USE-флаг `"symlink"`. Например, предварительно выполнив:

```
echo sys-kernel/vanilla-sources symlink >> /etc/portage/package.use/custom
```

При переходе к более крупной версии ядра, например 3.20 4.0.0, часто возникает необходимость посмотреть перечень изменений между настройками ядра (`make oldconfig`). Для этого выполните:

```
cl-kernel --kver 4.0.0 --kver-old=3.19.1
```

Настройка `calculate-sources`

На примере `vanilla-sources` мы научились устанавливать и собирать различные пакеты ядра. Но как быть, если нужно поставить обновление ядра "на поток" с вашими изменениями настроек и патчами? Нет ничего проще!

1. Сбросьте у ядра USE-флаг `"minimal"`:

```
echo sys-kernel/calculate-sources -minimal >>
/etc/portage/package.use/custom
```

2. Установите исходники ядра без компиляции:

```
USE="-vmlinuz" emerge sys-kernel/calculate-sources
```

3. Измените настройки:

```
cl-kernel
```

В последнем пункте нет ссылки на версию ядра, т.к. установка пакета переписала символьическую ссылку /usr/src/linux. Проверить это можно, выполнив:

```
cl-kernel --kver list
* 3.19.0
* 3.18.7-calculate *
```

Если вы не уверены, лучше указать ядро явно:

```
cl-kernel --kver=3.18.7-calculate
```

Обратите внимание, что напротив версии ядра стоит уже не красная, а зелёная звёздочка.

Пока ядро собирается, посмотрите на полученный шаблон настройки ядра: он будет содержать только внесённые вами изменения - отличия от оригинальной версии настроек ядра.

Пример шаблона после отключения поддержки ReiserFS:

```
cat /var/calculate/templates/kernel/10-calculate-x86_64-3.18
# Calculate format=kernel name=.config
os_install_arch_machine==x86_64&&merge(sys-kernel/calculate-sources)>=3.18
!CONFIG_REISERFS_FS=m
```

Обратите внимание, что повторное выполнение `cl-kernel` учитывает внесённые вами изменения. Чтобы сбросить их, удалите созданный вами шаблон.

Оптимизация ядра

Использование патчей

Для разработчиков

Интерактивная сборка системы

Введение

Интерактивная сборка - это новый подход в создании своего собственного загрузочного образа. Вы можете собирать необходимые пакеты, менять настройки и при этом видеть результат своей работы, сразу же тестируя собираемый дистрибутив. При разработке нового метода сборки преследовались следующие цели:

- Позволить каждому желающему создавать свой дистрибутив системы в соответствии с его взглядами и потребностями;
- Сделать процесс сборки системы более легким и доступным.

Как работает Calculate Builder

Использование интерактивного режима сборки доступно во всех дистрибутивах Calculate Linux начиная с версии 9.8. Для использования режима сборки воспользуйтесь режимом загрузки Builder на USB Flash или LiveCD.

Во время загрузки в Builder-режиме файловая система монтируется из трех слоев `aufs2`:

- Первый слой, *calculate*, представляет собой `livecd.squashfs`-образ системы, загружаемый с носителя и примонтированный в режиме "только для чтения". Он берётся за основу будущего дистрибутива.
- Второй слой - *delta* - слой, в котором будут сохраняться все изменения во время сборки нового дистрибутива.
- Третий слой, *workspace*, - рабочий слой, в котором Вы производите все изменения над исходной системой.

После загрузки все три слоя будут доступны в директории `/mnt/scratch`.

Вы можете запускать программы, менять настройки, создавать файлы - все ваши изменения будут сохраняться в слое `workspace`, не внося изменений в итоговый образ нового дистрибутива.

Интерактивная сборка происходит в директории `/mnt/builder`, являющейся результатом объединения двух слоев - `calculate` и `delta`. Вы также можете видеть все происходящие изменения, выполняя в процессе сборки необходимое тестирование собираемых приложений.

Процесс сборки системы

В пакет `calculate-builder` входит утилита `cl-builder`, которая используется для перехода в интерактивный режим сборки. Выполните `cl-builder` для подготовки системы к сборке. После выполнения команды приглашение в командной строке изменит свой цвет на коричневый (цвет может быть другим в зависимости от типа терминала) и вы окажетесь в chroot-окружении `/mnt/builder`. Директории `/proc`, `/dev`, `/dev/pts`, `/usr/calculate/share` базовой системы будут примонтированы автоматически, а также перенесён файл `resolv.conf`. Таким образом, сразу после выполнения `cl-builder` можно приступить к изменениям системы. Вы можете обновить дерево портей (команда `cl-update --sync-only`), а также обновлять, устанавливать или удалять программы. Результат установки программ будет отражаться и на загруженной системе. При этом все ваши действия в загруженной системе не затронут `/mnt/builder` и останутся только в слое `workspace`. Для избежания конфликтов в работе программ перед установкой пакетов всегда выполняйте команду `cl-builder`.

По завершении сборки выйдите из chroot-окружения, набрав в консоли `exit` либо нажав комбинацию клавиш `Ctrl+D`.

Шаблоны установки

Шаблоны - это конфигурационные файлы, в которых хранятся изменения настроек программ. Шаблоны могут содержать условные блоки, а также внутренние переменные для более гибкой настройки системы. Шаблоны утилит Calculate хранятся в директории `/usr/share/calculate/templates`. По аналогии с ними вы можете создать свои шаблоны в директории `/var/calculate/templates`.

Сохранение внесенных изменений

После того как вы закончили работу над изменениями текущего дистрибутива и вышли из chroot-окружения, вы можете создать загрузочный образ LiveCD, включающий все внесённые изменения. Для этого воспользуйтесь командой

```
cl-image iso
```

Загрузочный образ будет создан в файле с расширением `.iso` в директории `/var/calculate/linux`.

Если вы загружались с CD либо USB-Flash, то для всех действий может не хватить оперативной памяти компьютера. Чтобы избежать этого, примонтируйте свободный раздел жесткого диска либо сетевого диска в директорию `/var/calculate/linux`.

При загрузке с USB Flash вы можете сохранить все изменения в файле `livecd.squashfs` на вашей флешке. К концу файла будет добавлен порядковый номер сборки. При следующей загрузке будет использован новый образ со всеми изменениями. При последующих сборках старые файлы с образами будут удалены.

Установка системы

Полученный в результате изменений текущей системы ISO образ на 100% совместим с Gentoo и обладает всеми свойствами Calculate Linux. Систему можно загрузить с LiveCD, установить на жесткий диск, записать на USB Flash либо переносной USB-HDD. Возможность модификации полученного дистрибутива с помощью загрузки в Builder-режиме сохраняется. Таким образом, вы можете неограниченное число раз менять состав пакетов обычным для Gentoo образом - через обновление дерева портежей.

Примеры

Добавление в дистрибутив CLS браузера Opera, используя загрузочный CD

Выполните следующие шаги:

1. загрузитесь с CD в режиме Builder
2. выполните в терминале: `cl-builder`
3. убедившись что цвет курсора изменился, установим браузер командой: `emerge opera`
4. выйдем из chroot, набрав `exit`
5. при необходимости подмонтируем свободный раздел жёсткого диска: `mount /dev/sdaX /var/calculate`
6. сохраним изменения в новом файле с ISO образом: `cl-image iso`

Установка CLS на флешку и обновление дерева портежей

Для выполнения этой операции на компьютере должно быть установлено не менее 2 Гб оперативной памяти, т.к. на обновление дерева портежей может потребоваться достаточно большое количество памяти.

Выполните следующие шаги:

1. загрузитесь с CD в обычном режиме
2. установите систему на флешку: `cl-install -d /dev/sdX1` (вместо sdX1 укажите необходимое устройство, например, `sdb1`)
3. перезагрузите компьютер, выбрав загрузку с флешки, и выберите в меню загрузки режим Builder
4. выполните в терминале команду `cl-builder`
5. убедившись, что курсор изменил цвет, обновите дерево портежей, выполнив `cl-update --sync-only`
6. выйдем из chroot, набрав `exit`
7. обновите файл `livecd.squashfs`, выполнив: `cl-image squash`
8. перезагрузите компьютер

При недостаточном объеме оперативной памяти следует установить CLS на жесткий диск в режиме Builder, тогда все изменения будут кэшироваться на жестком диске. Команда `cl-image squash` при этом будет не доступна, а результат работы можно получить в виде готового ISO образа, при помощи команды `cl-image iso`.

Системные утилиты

- [Системные утилиты](#)
- [Управление ПО](#)
- [Поиск программ](#)
- [Установка и удаление программ](#)
- [Исправление зависимостей](#)
- [Настройки программ](#)
- [Управление сервисами](#)
- [Содержимое пакета](#)
- [Обеспечение безопасности](#)

- [Полезное](#)
- [Зависимости пакетов](#)
- [Экономия трафика](#)
- [Чистка distfiles](#)
- [Дефрагментация дисков](#)
- [Проверка жесткого диска](#)

Управление ПО

Поиск программ

Для быстрого поиска программ служит программа `eix` (пакет `app-portage/eix`), имеющая собственную базу данных для ускорения поиска.

Пример:

```
eix mozilla  
eix -S browser
```

Отобразить список установленных пакетов можно при помощи команды:

```
qlist -I
```

Отобразить список установленных пакетов с версией:

```
qlist -Iv
```

Отобразить список установленных пакетов с версией и USE флагами:

```
qlist -Uv
```

Если вы не нашли интересующей вас программы в дереве портежей, вы можете поискать ее в оверлеях. Для поиска воспользуйтесь сайтом <http://gpo.zugaina.org>. Название оверлея будет справа в нижней строке результата поиска.

Прежде чем установить программу, найденную на сайте, подключите оверлей в вашу систему, выполнив:

```
layman -a <оверлей>
```

Список оверлеев можно получить командой:

```
layman -L
```

Установка и удаление программ

Для [установки и удаления программ](#) используйте программу `emerge` (`sys-apps/portage`).

```
emerge kde-base/kgoldrunner  
emerge -C mc
```

В приведенном примере будет установлена игра `kgoldrunner` и удалена программа `mc` (Midnight Commander).

В случае если программа замаскирована, вы можете воспользоваться опцией `--autounmask` для вызова справки по размаскировке пакетов.

Пример установки замаскированной версии пакета:

```
emerge --autounmask =www-client/opera-11.10.2092
```

В конце перечня пакетов, подлежащих установке, вы увидите следующие рекомендации:

```
The following keyword changes are necessary to proceed:  
#required by =www-client/opera-11.10.2092 (argument)  
=www-client/opera-11.10.2092 ~amd64
```

Поместите текст под фразой "The following keyword changes are necessary to proceed:" в файл /etc/portage/package.keywords/custom (комментарии можно опустить).

```
echo "=www-client/opera-11.10.2092 ~amd64" >>  
/etc/portage/package.keywords/custom
```

Есть и более простой способ подтверждения размаскировки - использовать dispatch-conf (см. ниже).

Исправление зависимостей

Когда вы вносите изменения в установленные пакеты, может наступить случай нарушения зависимостей. Чтобы выявить и исправить такие нарушения, мы рекомендуем после обновления или удаления какой-либо программы выполнять команду revdep-rebuild.

Пример:

```
revdep-rebuild
```

Настройки программ

Во время обновления программ, для предотвращения ошибок, новые файлы настроек создаются с суффиксом `.cfg0000`. Для некоторых сервисов важно после обновления заменять старые настройки новыми. Иногда старые настройки могут привести в нерабочее состояние установленную программу, например, старые скрипты запуска сервисов, расположенные в директории /etc/init.d. Для своевременной замены конфигурационных файлов используйте программу dispatch-conf.

Пример:

```
dispatch-conf
```

После запуска программа будет показывать отличия новой версии каждого конфигурационного файла от старого. Обратите внимание: если настройки файла были модифицированы Calculate (вы увидите соответствующий комментарий), заменять эти настройки не следует - нажмите клавишу `z` для пропуска изменений. Для замены файла новым нажмите клавишу `u`.

Управление сервисами

Для добавления и удаления скрипта из уровня запуска служит скрипт rc-update.

Примеры:

```
# вывести список сервисов  
rc-update show  
# добавить numlock на уровень запуска default  
rc-update add numlock default  
# перестать запускать numlock  
rc-update del numlock
```

Показать запущенные сервисы можно командой:

```
rc-status --all
```

Содержимое пакета

Получить перечень файлов установленного пакета, а также узнать, какому пакету принадлежит файл в системе, можно при помощи программ qlist и qfile (app-portage/portage-utils).

Пример:

```
# список файлов пакета app-portage/portage-utils
qlist -ae app-portage/portage-utils
# узнать пакет в который входит qfile
qfile qfile
```

Вам также может пригодиться программка `which` (`sys-apps/which`), показывающая путь к файлу.

Пример:

```
# which equery
/usr/bin/equery
```

Обеспечение безопасности

Постоянное обновление системы - одно из важнейших мероприятий по обеспечению безопасности.
Можно следить за обновлениями посредством GLSA - `glsa-check` (`app-portage/gentoolkit`).

Просмотреть пакеты, которые необходимо переустановить:

```
glsa-check -p $(glsa-check -t all)
```

Переустановка уязвимых пакетов:

```
glsa-check -f $(glsa-check -t all)
```

Не помешает после этой операции проверить зависимости, см. выше *Исправление зависимостей*.

Полезное

Зависимости пакетов

Получить информацию о зависимостях пакетов можно с помощью программы `qdepends` (`app-portage/portage-utils`).

Пример:

```
qdepends -Q python
```

Экономия трафика

Если у Вас дорогой или лимитированный трафик, то можно сэкономить с помощью пакета `getdelta`.

Для его установки, если у вас архитектура *i686*, выполните:

```
ACCEPT_KEYWORDS=~x86 emerge getdelta
```

если *_x86_64_*, то:

```
ACCEPT_KEYWORDS=~amd64 emerge getdelta
```

Необходимо добавить в файл `/etc/make.conf` строку:

```
FETCHCOMMAND="/usr/bin/getdelta.sh \$URI \$DISTDIR/\$FILE"
```

Далее все действия не отличаются от обычной установки программ, с той лишь разницей, что качаются дельты исходников, что значительно сокращает объем необходимого на обновление трафика.

Чистка distfiles

С течением времени может накопиться достаточно большое количество разных версий пакетов программ. При нехватке дискового пространства вы можете удалить пакеты с устаревшими версиями программ, для этого воспользуйтесь программой eclean-dist (app-portage/gentoolkit):
`eclean-dist`

Дефрагментация дисков

Современные файловые системы, используемые в Linux, минимизируют фрагментацию дисков, поэтому существует достаточно небольшое количество программ для дефрагментации.

Для файловой системы XFS - пакет sys-fs/xfsdump (утилиты для дефрагментации и настройки XFS).

Пример команды для просмотра текущей фрагментации диска:

```
xfs_db -r -c frag /dev/sdax
```

где X - номер раздела.

Пример команды для дефрагментирования:

```
xfs_fsr -v /dev/sdax
```

Проверка жесткого диска

Для проверки жесткого диска на битые секторы используйте утилиту badblocks.

```
badblocks -svn -o /sda_log.txt -b 4096 -c 256 -p 2 /dev/sda
```

Оптимизация системы

- [Оптимизация системы](#)
- [Ускорение загрузки](#)
- [Ускорение запуска приложений](#)
- [prelink](#)
- [preload](#)
- [Повышение производительности](#)
- [Уменьшение размера дистрибутива](#)
- [Удаление неиспользуемых языков](#)
- [Высвобождение дополнительного места](#)

Ускорение загрузки

Calculate Linux использует для загрузки OpenRC. Благодаря переходу на бинарные утилиты загрузки и дополнительной оптимизации производительность существенно выросла, система стала загружаться значительно быстрее. В OpenRC предусмотрен режим параллельной загрузки, которая теперь включена по умолчанию в файле /etc/rc.conf:

```
rc_parallel="YES"
```

Ускорение запуска приложений

prelink

prelink - механизм предварительного связывания пакетов. Выполняется при сборке всех дистрибутивов Calculate Linux. В CLD/CLDX 9.9 prelink был добавлен в cron и теперь выполняется по расписанию. Включается в файле /etc/conf.d/prelink опцией:

preload

Начиная с версии 9.9, Calculate Linux Desktop включает утилиту `preload`. Работа программы становится заметной спустя некоторое время. Стартуя в *boot*-уровне загрузки, демон на протяжении всей работы компьютера анализирует время работы приложений. Впоследствии он уже сам подгружает необходимые программы и библиотеки в кэш памяти.

Благодаря тому, что `preload` занимает кэш дисковой системы, это не препятствует обычной работе - более того, ускоряет не только запуск отдельных приложений, но и загрузку системы в целом, создавая подобие параллельной загрузки.

Повышение производительности

Дистрибутивы Calculate Linux собраны с флагом компиляции, позволяющим запускать систему на любых [i686](#) и [x86_64](#)-совместимых процессорах. Вы можете несколько улучшить производительность вновь собранных пакетов, раскомментировав новые значения переменных `CFLAGS` и `CXXFLAGS` в файле `/etc/make.conf`:

```
CFLAGS="-O2 -march=native -pipe"
CXXFLAGS="${CFLAGS}"
```

Для повышения производительности системы в целом необходима пересборка всех пакетов.

```
emerge -e system
emerge -e world
```

Будьте внимательны, пересборка пакетов может потребовать определенного опыта, в случае если придется разрешать зависимости во время сборки. При выполнении будут загружены и скомпилированы все пакеты программ, входящие в дистрибутив. На выполнение может потребоваться от 5 часов и более, в зависимости от дистрибутива и производительности компьютера.

Уменьшение размера дистрибутива

Удаление неиспользуемых языков

Calculate Linux собирается с поддержкой нескольких языков. Тем не менее после установки системы в переменной `LINGUAS` файла `/etc/make.conf` можно оставить только нужный язык или языки.

Пример:

```
LINGUAS="en ru"
```

Для удаления из системы неиспользуемых языковых файлов потребуется обновление, которое будет заключаться в пересборке большого количества пакетов. Обратите внимание, что, если вы переопределите данную переменную в `/etc/make.conf`, все затронутые бинарные пакеты отныне всегда будут компилироваться.

Высвобождение дополнительного места

Если вы готовите систему к установке и планируете в дальнейшем обновлять только из бинарных пакетов собственной сборки, вы можете существенно сократить объем LiveCD-образа, удалив исходники ядра, портежи и, возможно, сам компилятор `gcc`. Вы можете выиграть в объеме, удалив:

- Portage : 35Mb в образе и 513Mb после установки
- исходники ядра : 76Mb в образе и 367Mb в установленной системе

Загрузка модулей ядра

Формат OpenRC

Для управления автоматической загрузкой модулей ядра в Gentoo используется файл `/etc/conf.d/modules`. В этом файле можно определить список модулей как для определенной версии ядра, так и для релиза ядра или основной версии ядра. Причем в автозагрузке используется только один параметр, точнее указывающий на версию версии ядра.

Например, если есть параметры:

```
modules_3="mpref"
modules_3_6="acpi-cpufreq"
```

то для ядра 3.6.x будет загружен модуль `acpi-cpufreq`, для всех остальных ядер 3.x будет загружен `mpref`. Использование только одного параметра затрудняет управление списком загружаемых модулей в зависимости от параметров системы, состава пакетов.

Система загрузки модулей в Calculate Linux

Для удобного управления загрузкой модулей ядра в `/etc/conf.d/modules` добавлен код, загружающей модули из конфигурационных файлов в директории `/etc/modules-load.d`. Используя шаблоны можно создавать конфигурационные файлы со списками в зависимости от версии ядра, версии определенных пакетов и т.д. Например шаблонами оверлея при установке пакета `virtualbox-modules` в автозагрузку будут добавлены модули `vboxdrv`, `vboxnetflt` и `vboxnetadp` в файле `/etc/modules-load.d/virtualbox.conf`.

Формат файлов

Конфигурационные файлы описывающих модули находятся в директории `/etc/modules-load.d` и имеют расширение `".conf"`. Файлы состоят из списка имен модулей ядра, разделенных переводом строки. Пустые строки и строки начинающиеся с `"#"` или `";"` игнорируются. Списки не поддерживают указание параметров модулей ядра - для указания параметров используйте `/etc/modprobe.d`.

Миграция на новый формат

При выполнении обновления `eix-sync` будет создан файл копии старых настроек `/etc/conf.d/modules.old`, пользовательские модули будут перенесены в файл `/etc/modules-load.d/migrate.conf`. В процессе миграции будут пропущены модули управления частотой процессора, т.к. они подгружаются автоматически при изменении параметров частоты.

Ревизии системы

- [Ревизии системы](#)
- [Формирование состава пакетов](#)
- [Управление world-файлом при помощи шаблонов](#)
- [Ревизии системы](#)

Формирование состава пакетов

По мере установки новых программ формируется системный файл (`/var/lib/portage/world`), содержащий список установленных пакетов. Файл содержит неполный список пакетов, установленных в системе. В нём отсутствуют пакеты, вычисляемые по зависимостям, например библиотеки.

В качестве примера, установим электронные таблицы Gnumeric, выполнив:

```
emerge -a gnumeric
```

при выполнении пакетный менеджер может предложить установить несколько пакетов - сам `app-office/gnumeric`, а так же необходимые для работы пакеты `gnome-extra/libgsf` и `x11-libs/goffice`. В `world`-файл будет добавлен только пакет `app-office/gnumeric`, - пакет, который вы указали в качестве параметра [emerge](#).

При удалении Gnumeric:

```
emerge -C gnumeric
```

пакет `app-office/gnumeric` будет удалён из `world`-файла, при этом все зависимые пакеты останутся в системе.

Для удаления зависимостей, выполните:

```
emerge -ac
```

При выполнении этой команды сформируется дерево пакетов исходя из списка в `world`-файле, с включением зависимостей и, в случае наличия в системе установленных и не связанных пакетов, будет предложено их удалить.

Управление `world`-файлом при помощи шаблонов

Начиная с 13-й версии Calculate Linux изменилась система управления составом пакетов дистрибутива. В предыдущих версиях системы дерево зависимостей формировалось через так называемые мета-пакеты, содержащие только необходимые зависимости. Такой подход имел свои плюсы и минусы. В любой момент новая версия одного из мета-пакетов могла включить новую программу или исключить либо принудительно удалить другую. Мета-пакеты также успешноправлялись с возможными блокировками, которые могут возникнуть при формировании дерева зависимостей. Основной недостаток же был в сложности удаления предустановленных программ пользователем.

Сейчас помимо пакетного менеджера, в `world`-файл могут вносить изменения утилиты Calculate. Это позволило избавиться от мета-пакетов. Во время выполнения синхронизации дерева портей и оверлея, содержащего шаблоны утилит Calculate командой `eix-sync`, происходит вызов `cl-core`, который в свою очередь обрабатывает [шаблоны](#).

При помощи шаблонов формируются версии состава пакетов дистрибутива. К примеру, в случае замены одной программы на другую, вторая будет записана в `world`-файл вместо первой. Шаблоны позволяют гибко формировать `world`-файл, учитывая наличие заменяемой программы в системе.

Версию `world`-файла можно посмотреть в `/etc/calculate/ini.env`, пример:

```
[update]
world = 22
```

Шаблоны формирования `world` на данный момент можно посмотреть здесь:

```
/var/lib/layman/calculate/profiles/templates/3.1/6_ac_update_sync/world/
```

Шаблоны делятся на две группы:

- Формирование `world`-файла с нуля;
- Обновление текущего `world`: удаление, замена, добавление пакетов;

Шаблоны разделяются на категории, такие как `graphics`, `network`, `multimedia`, и т.д., в каждой из них перечислены необходимые для дистрибутива пакеты.

Шаблоны разделены на версии таким образом, что в каждом из них описаны правила обновления с предыдущей версии, пример:

```
#удалить geeqie в CLD
#?os_linux_shortname==CLD#
!media-gfx/geeqie
```

```
#os_linux_shortname#  
  
# добавить catfish в CLDX  
#?os_linux_shortname==CLDX#  
dev-util/catfish  
#os_linux_shortname#  
  
# заменить chromium на firefox  
#?pkg(www-client/chromium)!=#  
www-client/firefox  
#pkg#
```

Обновить world-файл можно выполнить и вручную, при помощи команды:

```
cl-update --update-rev
```

Заново сформировать world-файл, без учёта установленных программ:

```
cl-update --rebuild-world
```

Сформировать world-файл исходя из лога установленных программ, без учёта предустановленных программ:

```
regenworld
```

Ревизии системы

Изменения в формировании файла world также повлияли на систему исправления ошибок/настроек в дистрибутиве. Ранее исправление ошибок происходило при обновлении пакетов и имело следующие недостатки:

- создание новых ревизий пакетов (чаще мета пакетов), для исправления во время полного обновления системы
- переустановка мета пакетов вызывала повторное наложение шаблонов для исправления
- создание ревизий для пакетов, не связанных с этой ошибкой, если исправление нужно внести до выполнения обновлений определенных пакетов
- ошибка не будет исправлена если не будет переустановлен необходимый пакет

Новое исправление ошибок и настроек запускается командой `cl-update --update-rev` и не привязана к конкретным пакетам. Команда запускается с обновлением `world` файла, при выполнении `eix-sync`.

Все исправления содержатся в шаблонах `6_ac_update_sync/revision`. Исправление может быть как скриптом, так и указанием списка пакетов, которые нужно перенастроить. Каждое исправление привязано к конкретной ревизии.

Установленный дистрибутив содержит ревизию системы в файле `/etc/calculate/ini.env`, поэтому одни и те же исправления не будут применены несколько раз.

Еще одним преимуществом такого подхода является то, что обновление рабочих систем и подготовка образа дистрибутива происходит по одной схеме, что позволяет уменьшить количество ошибок при обновлении системы, не проявляющихся в стейджах.

calcboot

- [calcboot](#)
- [Назначение](#)
- [Описание модуля calcmenu.c32](#)
- [1. запоминание значений параметров](#)

- [2. замещение значений сохраненных параметров](#)
- [3. замещение значений параметров по позициям](#)
- [4. восстановление начальной позиции курсора в меню по параметру](#)
- [5. указание значений по умолчанию для параметров \(добавлено в версии 3.86.5\)](#)

Назначение

`calcboot` представляет собой модуль для Syslinux (`calcmenu.c32`), позволяющий запоминать выбор при использовании нескольких загрузочных меню. Также в нем хранятся фоновые изображения для Grub и Syslinux.

Описание модуля `calcmenu.c32`

`calcmenu.c32` основан на `vesamenu.c32` и реализует как все его возможности, так и дополнительные:

1. запоминание значений параметров

При использование нескольких меню в поле APPEND заносится имя файла меню, к которому нужно перейти при выборе данного пункта. В отличии от стандартного `vesamenu.c32` указанные параметры для меню не сбрасываются.

Пример пункта меню:

```
APPEND isolinux.cfg calculate=,be_BY
```

При выборе такого пункта меню будет загружено меню из файла `isolinux.cfg`, а параметр `calculate=, be_BY` будет сохранен. В итоге при выборе пункта меню из `isolinux.cfg`, содержащего

```
KERNEL /boot/vmlinuz
APPEND root=/dev/ram0 initrd=/boot/initrd init=/linuxrc looptype=squashfs
unionfs
```

будет запущено ядро с параметрами, указанными в APPEND, плюс `calculate=, be_BY`.

2. замещение значений сохраненных параметров

Сохраненные параметры не повторяются. То есть если параметр `calculate` уже сохранен, и осуществляется выбор пункта меню, у которого также указан параметр `calculate`, значение `calculate` будет заменено на новое.

3. замещение значений параметров по позициям

Каждый параметр может хранить несколько значений, если они разделены запятой. Например, параметр `calculate` будет хранить язык и временную зону (`calculate=ru_RU,Europe/Moscow`). Для реализации этого создаются два различных меню (`lang.cfg`, `timezone.cfg`), пункты меню которых содержат следующие APPEND:

```
lang.cfg
...
APPEND timezone.cfg calculate=ru_RU,
```

```
timezone.cfg
...
APPEND othermenu.cfg calculate=,Europe/Moscow
...
```

При выборе таких пунктов меню для `othermenu.cfg` будет сохранен параметр `calculate` со значением `ru_RU,Europe/Moscow`.

4. восстановление начальной позиции курсора в меню по параметру

Чтобы запоминалось положение курсора выбора языка, временной зоны и т.д., в файле описания меню используется

`MENUPARAM` параметр номер

где "параметр" - название параметра, который хранит выбранное значение, а "номер" - номер позиции в значении. Например, для `timezone.cfg`, хранящей описание меню с часовыми поясами, используется `MENUPARAM calculate 1`

т.е. параметр `calculate`, второе значение. Таким образом, при отображении меню `timezone.cfg`, если есть сохраненный параметр `calculate` и у него есть значение во второй позиции, то курсор будет указывать на необходимый пункт меню.

5. указание значений по умолчанию для параметров (добавлено в версии 3.86.5)

Для указания значений используется

`DEFAULTPARAM` параметры

Например, для указания русского языка по умолчанию:

`DEFAULTPARAM calculate=lang:ru_RU, keymap:ru_RU`

calckernel

Назначение

`calckernel` --- это инструмент, позволяющий автоматизировать компиляцию ядра и создание Ramdisk.

Отличие от genkernel

`calckernel` добавляет в Ramdisk ряд возможностей:

- реализация загрузки в режиме [интерактивной сборки](#)
- поддержка `aufs` и `unionfs` для LiveCD
- загрузка образа liveCD в память с автоувеличением размера `tmpfs`
- отмонтирование CD при успешной загрузке образа в память
- использование KMS (Kernel Mode Setting)
- использование `udev` для автоматической загрузки необходимых модулей
- автоматическое определение видео драйвера
- использование параметра ядра `Calculate` для загрузки определенного видео драйвера

calculate-sources

Пакет `sys-kernel/calculate-sources` отличается других пакетов '`*sources`' из портежей тем, что позволяет автоматизировать сборку ядра Linux:

- наложение патчей при помощи шаблонов;
- создание конфигурации при помощи шаблонов;
- компиляция ядра и модулей;

- установка ядра в систему

Используемые USE флаги

Для управления порядком сборки используются следующие USE флаги:

- при включенном USE флаге 'vmlinuz' во время сборки пакета ядро и модули будут скомпилированы и помещены в /boot и /lib соответственно.
- флаг 'minimal' используется для удаления из системы исходного кода собранного ядра, при этом в исходниках остаются только файлы, необходимые для сборки сторонних модулей ядра.

Для сборки ядра и удаления исходного кода ebuild использует различные версии calculate-kernel e-классов.

Патчи на ядро

До марта 2013 патчи хранились в архивах calculate-sources-3.X.tar.bz2, что приводило к тому, что при обновлении одного из патчей (например при обновлении подверсии ядра) приходилось создавать новые архивы с дополнительными номерами (версии, ревизии и т.д.) и усложнять ebuild-ы. В связи с изменением способа наложения патчей во время сборки пакета, появилась возможность перенести патчи в шаблоны утилит Calculate, получив дополнительные преимущества:

- разделять патчи на десктопные и серверные;
- использовать разные патчи для разных версий ядра Caluclate-sources использует следующие основные патчи:
- aufs3 - файловая система, производящая каскадно-объединённое монтирование других файловых систем, используемая для запуска liveCD;
- fbcondecor - декорации в консоли и дополнительные, используемые в десктопных системах;
- TuxOnIce - используется для ускорения "засыпания" и "пробуждения" компьютера;
- BFQ - I/O планировщик, созданный как замена CFQ (и основан на его коде), основная мысль -- более честное разделение I/O между процессами.

Пользовательские патчи

Выполнение патчей происходит при событии компиляция пакета с действием ac_install_patch относительно пути S ("\${WORKDIR}/{\$P}"), фильтрация по пакетам и версиям осуществляется через функцию merge(), формат шаблонов обычно diff (представляющие обычные патчи). Таким образом в простейшем варианте необходим файл патча со следующим заголовком:

```
# Calculate format=diff install.ac_install_patch==on&&merge(sys-
kernel/calculate-sources)>=версия
```

...
содержимое патча
...

Конфигурация ядра

До изменения системы наложения патчей конфигурация ядра хранилась в оверлее profiles/kernel в виде готового конфигурационного файла как для сервера, так и для десктопа. В связи с изменением метода наложения патчей на пакет изменился и способ создания конфигурационного файла, что дало ряд преимуществ:

- нет разделения настроек на десктопные и серверные ядра (серверное ядро получается из конфигурации ядра, выполнением изменений по шаблону);
- параметры ядра, которые необходимо изменить при наличии патчей вынесены в отдельные шаблоны;
- возможность менять параметры ядра, патчи и маскировку не затрагивая ebuild-файлы

Пользовательские параметры ядра

Так как создание конфигурационного файла происходит при сборке пакета, то для изменения параметров ядра необходимо создать файл с измененными параметрами, связанный с действием `ac_install_patch`. Например, чтобы включить в ядро файловые системы EXT2, EXT3 и EXT4 можно создать следующий файл:

```
# Calculate name=.config format=openrc
install.ac_install_patch==on&&merge(sys-kernel/calculate-sources)>=версия
CONFIG_EXT2_FS=y
CONFIG_EXT3_FS=y
CONFIG_EXT4_FS=y
```

Порядок маскировки ядер

Если ядро находится в `hard` маске, то это означает что оно еще не готово для полноценного использования: не все необходимые патчи, не готова конфигурация, в ходе эксплуатации выявлены критичные ошибки, ошибки во время сборки. После решения выше перечисленных задач с ядра снимается `hard` маска, и оно остаётся в маске по `keywords`, пока не будет использовано в одном из стейджей.

Имена ядер

При установке ядра из пакета `calculate-sources` с включенным USE-флагом `vmlinuz` происходит компиляция и установка ядра в директорию `/boot`. К именам файлов `vmlinuz`, `config`, `initramfs` и `System.map` дописывается окончание, состоящее из версии ядра, архитектуры и сокращенного имени дистрибутива (пример: "-3.9.7-i686-CLD"). В случае наличия таких файлов в директории `/boot`, к старым версиям дописывается окончание ".old".

На ядро и вспомогательные файлы создаются символические ссылки. Для того чтобы не модифицировать загрузчик, имена символьических ссылок не подлежат изменению. Символьические ссылки были необходимы для Legacy Grub. Grub 2 сам автоматически определяет используемые файлы для формирования своего конфигурационного файла. В ближайших версиях символические ссылки и `kernel_uid` будут удалены.

При необходимости, сгенерировать настройки Grub можно командой:

```
cl-setup-boot
```

Удаление старых версий ядра

Старые версии ядра удаляются при удалении соответствующих версий пакета `calculate-sources`. Поэтому, если вы не хотите оставлять их в системе, убедившись, что новое ядро работает нормально, удалите ненужные пакеты:

```
emerge -c
```

Если же необходимо удалить только исходный код ядра, оставив собранное ядро, модули и конфигурационный файл - выключите автоматическое удаление в файле `/etc/calculate/ini.env`

```
[update]
remove_kernel = off
```

keyexec

Описание

keyexec предназначен для доступа терминального клиента к удаленному рабочему столу Windows. Программа работает через rdesktop-клиент.

При запуске программа будет передавать в rdesktop введенный пользователем при авторизации пароль: таким образом, использование Windows-приложений в линуксе будет происходить без дополнительных сложностей для пользователя.

За хранение паролей отвечает *служба хранения ключей ядра Linux*. Более подробно это описано [здесь](#).

Для работы программы необходимо наличие модуля [pam_keystore](#) в авторизации PAM, который сохраняет пароль пользователя в ключах ядра.

Формат запуска

```
keyexec rdesktop[0-9] <параметры>
```

Примеры

Запуск рабочего стола Windows-сервера с IP 192.168.0.1:

```
keyexec rdesktop4 "-p - 192.168.0.1"
```

Запуск рабочего стола Windows-сервера на 4-м рабочем столе, учетная запись берется из домена "calculate":

```
keyexec rdesktop4 "-d calculate -p - 192.168.0.1"
```

Запуск Adobe Photoshop в экране размером 1280x999, с оптимизацией передаваемого трафика:

```
keyexec rdesktop "-s 'C:\Program Files\Adobe\Adobe Photoshop CS3\photoshop.exe' -d calculate -5 -a 16 -g 1280x999 -p - -T 'Adobe Photoshop CS3' -S standard -zNDKE 192.168.0.1"
```

pam_keystore

Описание

_pam_keystore_ - pam-модуль предназначенный для хранения логина и пароля пользователя в "службе хранения ключей" ядра Linux.

Необходим для программы [keyexec](#). Используется при монтировании сетевых дисков пользователя при хранении учетных записей на сервере. Модуль входит в состав дистрибутива [Calculate Linux Desktop](#).

Использование

Дистрибутив Gentoo

Для подключения модуля измените файл /etc/pam.d/system-auth следующим образом:

```
auth      required      pam_env.so
auth      optional       pam_keystore.so use_first_pass
auth      sufficient    pam_unix.so use_first_pass
auth      required       pam_deny.so
```

добавив строку:

```
auth      optional       pam_keystore.so use_first_pass
```

Получение пароля пользователя из службы хранения ядра.

Пароль пользователя, который зашел в систему, может получить только пользователь *root*.

Свой пароль пользователь получить не может.

Чтобы получить пароль пользователя, необходимо пользователем *root* выполнить команду:

```
keyctl print $( keyctl request user user_name )
```

где

- *user_name* - имя пользователя, который вошел в систему.

Примечание: *keyctl* входит состав пакета *sys-apps/keyutils*.

3. Работа с Portage

1. [Введение в Portage](#)
2. [USE-флаги](#)
3. [Возможности Portage](#)
4. [Переменные среды](#)
5. [Файлы и каталоги](#)
6. [Настройка с помощью переменных](#)
7. [Смешение ветвей программного обеспечения](#)
8. [Дополнительные средства Portage](#)
9. [Отступление от официального дерева](#)
10. [Использование ebuild](#)

Введение в Portage

11. [Введение в Portage](#)
12. [Добро пожаловать в Portage](#)
13. [Дерево портежей](#)
14. [Сборочные файлы ebuild](#)
15. [Обновление дерева портежей](#)
16. [Обслуживание программного обеспечения](#)
17. [Поиск программ](#)
18. [Установка программ](#)
19. [Обнаружение документации к пакету](#)
20. [Удаление пакета](#)
21. [Обновление системы](#)
22. [Метапакеты](#)
23. [Когда Portage жалуется...](#)
24. [Слоты, виртуалы, ветви, архитектуры и профили](#)
25. [Блокировка пакетов](#)
26. [Маскировка пакетов](#)
27. [Отсутствие нужных пакетов](#)
28. [Неоднозначность названия пакета](#)
29. [Циклические зависимости](#)
30. [Ошибка извлечения](#)
31. [Защита системного профиля](#)

Добро пожаловать в Portage

Благодаря высокой гибкости и чрезвычайно богатым возможностям Portage можно признать одним из лучших средств управления программным обеспечением из существующих в Linux. Portage написана на Python и Bash.

Большинство пользователей взаимодействует с Portage с помощью команды emerge. Эта глава не призвана заменить страницу справки emerge. Просмотреть все возможные параметры команды emerge можно [здесь](#).

Дерево портежей

Сборочные файлы ebuild

Говоря о пакетах, мы часто имеем в виду программы, доступные пользователям Calculate через дерево портежей. Дерево портежей - это набор сборочных файлов ebuild, содержащих всю информацию, необходимую Portage для управления программным обеспечением (установки, поиска, извлечения и т.п.) По умолчанию сборочные файлы находятся в /usr/portage и /var/lib/layman/calculate. По второму пути находится оверлей Calculate, который также содержит сборочные ebuild файлы.

Когда Portage выполняет любые действия над пакетами программ, эти действия опираются на сборочные файлы, имеющиеся в системе. Поэтому необходимо регулярно обновлять сборочные файлы, чтобы Portage знал о новых программах, обновлениях, связанных с [безопасностью](#) и т.д.

Обновление дерева портежей

Дерево портежей обычно обновляется с помощью rsync. Обновление выполнить довольно просто, так как запуск rsync обеспечивается командой emerge:

```
emerge --sync
```

Если rsync выполнить невозможно из-за ограничений межсетевого экрана, дерево портежей все-таки можно обновить из ежедневных «снимков». Для автоматического извлечения и установки в системе новейшего снимка служит утилита emerge-webrsync:

```
emerge-webrsync
```

Важно вместе с деревом портежей выполнять обновление оверлея Calculate. Оверлей хранится в Git, его обновление можно выполнить утилитой layman:

```
layman -s calculate
```

Выполнить обновление сразу дерева портежей и оверлея Calculate, а также базы данных программы eix можно одной командой:

```
eix-sync
```

Обслуживание программного обеспечения

Поиск программ

Для поиска программ в дереве портежей по названию можно использовать встроенные возможности команды emerge. По умолчанию команда emerge --search выдает названия пакетов, соответствующих (как полностью, так и частично) заданному условию поиска.

Например, чтобы найти все пакеты, содержащие "pdf" в названии:

```
emerge --search pdf
```

Для поиска пакетов еще и по тексту описания можно использовать параметр --searchdesc (или -S):

```
# emerge --searchdesc pdf
Searching...
[ Results for search key : firefox ]
[ Applications found : 10 ]

* www-client/firefox
    Latest version available: 3.6.13
```

```
Latest version installed: [ Not Installed ]
Size of files: 59,577 kB
Homepage: http://www.mozilla.com/firefox
Description: Firefox Web Browser
License: || ( MPL-1.1 GPL-2 LGPL-2.1 )
...
```

Существуют вспомогательные средства для ускорения и автоматизации стандартных задач типа поиска по дереву портежей, формирования списка установленных пакетов, принадлежащих какой-либо категории и т.д. Для выполнения быстрого поиска пакета используйте утилиту eix. Пример поиска браузера firefox:

```
eix firefox
```

Пример поиска всех пакетов, в описании которых присутствует слово "browser".

```
eix -S browser
```

Установка программ

После того, как вы нашли нужное программное обеспечение, его можно легко установить с помощью команды emerge. Вот пример установки пакета gnumeric:

```
emerge gnumeric
```

Так как множество приложений зависит друг от друга, любая попытка установить какой-либо пакет программ может повлечь за собой также установку дополнительных пакетов. Не беспокойтесь, Portage справится с этим. Если вы захотите выяснить, что именно Portage собирается установить вместе с нужным вам пакетом, добавьте параметр -p (или --pretend). Например:

```
emerge -p gnumeric
```

После команды на установку пакета, Portage загружает из интернета необходимый исходный код (при необходимости), и по умолчанию сохраняет его в каталоге /var/calculate/remote/distfiles. После этого пакет распаковывается, компилируется и устанавливается. Если вы хотите, чтобы Portage только загрузил исходный код без его установки, добавьте к команде emerge параметр -f (или --fetchonly):

```
emerge -f gnumeric
```

Обнаружение документации к пакету

Многие пакеты содержат собственную документацию. Иногда USE-флаг doc определяет, следует ли устанавливать документацию к пакету. Проверить наличие USE-флага doc можно командой emerge -vp <название пакета>. Пример:

```
emerge -vpeselect
[ebuild R ] app-admin/eselect-1.2.11 USE="bash-completion -doc" 0 kB
```

USE-флаг doc можно включить или отключить как глобально в файле [/etc/portage/make.conf/custom](#), так и для отдельных пакетов, создав файл в директории /etc/portage/package.use и указать в нём флаг. В главе [USE-флаги](#) этот вопрос описывается более подробно.

Документация от вновь установленного пакета обычно находится в подкаталоге каталога /usr/share/doc, соответствующем названию пакета. Кроме того, можно вывести список всех установленных файлов утилитой equery. Примеры:

```
# ls -l /usr/share/doc/eselect-1.2.11
total 48
-rw-r--r-- 1 root root 296 Янв 26 21:36 AUTHORS.bz2
```

```
-rw-r--r-- 1 root root 14202 Янв 26 21:36 ChangeLog.bz2
-rw-r--r-- 1 root root 5320 Янв 26 21:36 developer-guide.txt.bz2
-rw-r--r-- 1 root root 3837 Янв 26 21:36 NEWS.bz2
-rw-r--r-- 1 root root 541 Янв 26 21:36 README.bz2
-rw-r--r-- 1 root root 703 Янв 26 21:36 release-guide.txt.bz2
-rw-r--r-- 1 root root 471 Янв 26 21:36 TODO.bz2
-rw-r--r-- 1 root root 2067 Янв 26 21:36 user-guide.txt.bz2
```

```
# equery feselect | less
/usr
/usr/bin
/usr/bin/bashcomp-config
/usr/bin/eselect
...
```

Удаление пакета

Когда вы захотите удалить пакет из системы, используйте команду emerge -C (или --unmerge). Это приведет к удалению из системы всех файлов, установленных пакетом, кроме конфигурационных файлов приложения, изменявшихся после установки. Сохранение конфигурационных файлов позволяет вернуться к работе с пакетом, если вы когда-нибудь решите снова его установить.

Внимание: Portage не проверяет, зависят ли другие пакеты от удаляемого! Однако вы получите предупреждение, если удаление пакета приведет к неработоспособности системы.

```
emerge -C gnumeric
```

После удаления пакета из системы, остаются пакеты, установленные по зависимостям. Чтобы Portage выявила все когда-то нужные пакеты, которые теперь можно удалить, используйте команду emerge -c (или --depclean). Мы вернемся к этому ниже.

Обновление системы

Чтобы система сохранялась в отличной форме (не говоря уже об установке свежайших обновлений, связанных с безопасностью), ее нужно регулярно обновлять. Так как Portage просматривает сборочные файлы только в локальном дереве портежей и оверлее, сперва потребуется обновить их. Обновив дерево портежей, вы сможете обновить систему командой emerge -u world. В следующем примере мы также пользуемся параметром -a (или --ask), который поручает Portage вывести список пакетов, которые она собирается обновить, и спросить вас, можно ли продолжать:

```
emerge -ua world
```

Portage будет искать более новые версии установленных приложений. Однако проверяется только версии приложений, явно установленных вами, а не тех, от которых они зависят. Если вы хотите обновить каждый пакет в системе, добавьте аргумент -D (или --deep):

```
emerge -uD world
```

Поскольку обновления, относящиеся к безопасности, случаются и в пакетах, которые были установлены по зависимостям, рекомендуется изредка запускать эту команду.

Если вы меняли какие-либо из USE-флагов, возможно, потом вы также захотите добавить параметр -N (или --newuse). Тогда Portage проверит, требует ли изменение установки новых пакетов или перекомпиляции существующих:

```
emerge -uDN world
```

Метапакеты

У некоторых пакетов в дереве портежей нет содержимого как такового, и они используются для установки набора других пакетов. Например, пакет kde полностью устанавливает среду KDE в вашей системе, привлекая различные KDE-пакеты в качестве зависимостей.

Если вы когда-либо захотите удалить из системы такой пакет, запуск emerge --unmerge не возымеет должного эффекта, так как пакеты, от которых он зависит, останутся в системе.

В Portage существует возможность удаления остаточных зависимостей, но так как зависимости программы меняются со временем, доступность программного обеспечения, прежде всего требуется полностью обновить всю систему, включая реализацию изменений, произведенных путем модификации USE-флагов. После этого можно запустить emerge -c (или --depclean), чтобы удалить остаточные зависимости. Когда это сделано, вам потребуется пересобрать приложения, ранее динамически связанные с удаленными пакетами, в которых они теперь не нуждаются.

Со всем этим управляются следующие три команды:

```
emerge -uDN world  
emerge -ca  
revdep-rebuild
```

Когда Portage жалуется...

Слоты, виртуалы, ветви, архитектуры и профили

Как уже сказано, Portage - чрезвычайно мощная система, поддерживающая множество возможностей, не хватающих другим системам управления программами. Чтобы это стало понятно, разберем несколько аспектов Portage, не вникая в подробности.

С помощью Portage разные версии отдельного пакета могут существовать в одной системе. В то время, как другие системы управления стремятся называть пакеты в соответствии с версией (например freetype и freetype2), в Portage используется технология **слотов** (SLOT), или областей. Пакет присваивает определенный слот своей версии. Пакеты с разными слотами способны существовать в одной системе. Например, у пакета freetype есть ebuild как со SLOT="1", так и со SLOT="2".

Существуют также пакеты, выполняющие одни и те же функции, но отличающиеся в реализации. Например metalogd, sysklogd и syslog-ng являются системными службами журналирования. Приложения, использующие "системный журнал", не могут зависеть от одной конкретной программы, например от metalogd, так как остальные программы ничем не хуже. В Portage предусмотрены виртуальные пакеты: каждая служба журналирования предоставляет virtual/syslog, и в результате в приложениях можно указывать зависимость от virtual/syslog.

Программное обеспечение может располагаться в различных ветвях дерева портежей. По умолчанию в системе разрешено только использование стабильных пакетов. Большинство новых программ при поступлении включаются в тестовую ветвь, что указывает на необходимость дополнительного тестирования перед тем, как включить их в стабильные. Хотя в дереве портежей и видны сборочные файлы для таких программ, Portage не станет обновлять их до тех пор, пока они не будут помещены в стабильную ветвь.

Некоторые программы имеются не для всех архитектур. Либо они не работают в определенных архитектурах, либо требуют дополнительного тестирования, или у разработчика нет возможности проверить, работает ли пакет в различных архитектурах.

Каждая установка Gentoo придерживается определенного *профиля*, который содержит, помимо прочего, список пакетов, необходимых для работоспособности системы.

Блокировка пакетов

Пример предупреждения о заблокированных пакетах (с --pretend)

```
[blocks in      ] mail-mta/ssmtp (is blocking mail-mta/postfix-2.2.2-r1)
```

```
!!! Error: the mail-mta/postfix package conflicts with another package.  
!!! both can't be installed on the same system together.  
!!! Please use 'emerge --pretend' to determine blockers.  
  
( !!! Ошибка: пакет mail-mta/postfix конфликтует с другим пакетом.  
!!! оба не могут находиться в системе одновременно. Пожалуйста,  
!!! запустите 'emerge --pretend' для выявления блокирующих пакетов. )
```

В файлах ebuild есть специальные поля, сообщающие Portage о зависимостях. Возможны два вида зависимости: зависимость сборки, объявленная в DEPEND, и зависимость выполнения, объявленная в RDEPEND. Когда одна из этих зависимостей явно указывает на несовместимость пакета или виртуального пакета, это вызывает блокировку.

Для разблокировки можно отказаться от установки пакета или предварительно удалить конфликтующий пакет. В данном примере можно отказаться от установки postfix или сначала удалить ssmtp.

Также возможно, что два пакета, подлежащие установке, блокируют друг друга. В этом редчайшем случае следует определить, зачем вам устанавливать оба пакета. В большинстве случаев можно обойтись одним.

Маскировка пакетов

Пример предупреждения о замаскированных пакетах:

```
!!! all ebuilds that could satisfy "bootsplash" have been masked.  
  
( !!! все сборки, удовлетворяющие "bootsplash", замаскированы.)
```

Пример предупреждения о замаскированных пакетах с указанием причины:

```
!!! possible candidates are:  
  
- gnome-base/gnome-2.8.0_pre1 (masked by: ~x86 keyword)  
- lm-sensors/lm-sensors-2.8.7 (masked by: -sparc keyword)  
- sys-libs/glibc-2.3.4.20040808 (masked by: -* keyword)  
- dev-util/cvsd-1.0.2 (masked by: missing keyword)  
- media-video/ati-gatos-4.3.0 (masked by: package.mask)  
- sys-libs/glibc-2.3.2-r11 (masked by: profile)  
  
( !!! возможные кандидаты:  
  
- gnome-base/gnome-2.8.0_pre1 (маскировка: ключ ~x86)  
- lm-sensors/lm-sensors-2.8.7 (маскировка: ключ -sparc)  
- sys-libs/glibc-2.3.4.20040808 (маскировка: ключ -*)  
- dev-util/cvsd-1.0.2 (маскировка: ключ отсутствует)  
- media-video/ati-gatos-4.3.0 (маскировка: package.mask)  
- sys-libs/glibc-2.3.2-r11 (маскировка: profile) )
```

Когда вы собираетесь установить пакет, не предназначенный для вашей системы, выдается ошибка маскировки. Нужно попытаться установить другую программу, существующую для вашей системы, или дождаться, пока пакет станет доступным. Всегда есть причина, по которой пакет замаскирован:

- **ключ ~arch**: пакет недостаточно проверен для помещения в стабильную ветвь. Подождите несколько дней или недель и попробуйте установить его еще раз.
- **ключ -arch** или **ключ -***: пакет не работоспособен в вашей архитектуре. Если вы полагаете, что он работает, сообщите об этом в bugzilla.

- **ключ отсутствует**: пакет еще не тестирулся в вашей архитектуре. Попросите группу портирования в архитектуру проверить пакет, или протестируйте его за них и сообщите о своих изысканиях в bugzilla.
- **package.mask**: обнаружено повреждение пакета, нестабильность или что-то худшее, и пакет заблокирован специально.
- **profile**: пакет считается не предназначенным для вашего профиля. В случае установки приложение может вызвать сбой системы или просто несовместимо с используемым профилем.

Отсутствие нужных пакетов

Пример предупреждения об отсутствии пакета:

```
emerge: there are no ebuilds to satisfy ">=sys-devel/gcc-3.4.2-r4".
!!!! Problem with ebuild sys-devel/gcc-3.4.2-r2
!!!! Possibly a DEPEND/*DEPEND problem.

( emerge: нет сборок, удовлетворяющих ">=sys-devel/gcc-3.4.2-r4".
!!!! Проблема с ebuild sys-devel/gcc-3.4.2-r2
!!!! Возможно, ошибка в DEPEND/*DEPEND. )
```

Приложение, которое вы пытаетесь установить, зависит от другого пакета, недоступного вашей системе. Пожалуйста, проверьте, есть ли такой запрос в [bugzilla](#), а если нет, сообщите об ошибке. Если вы не смешиваете ветви, такого не должно происходить, и это - явная ошибка.

Неоднозначность названия пакета

Пример предупреждения о повторяющихся именах ebuild:

```
!!!! The short ebuild name "aterm" is ambiguous. Please specify
!!!! one of the following fully-qualified ebuild names instead:

dev-libs/aterm
x11-terms/aterm

( !!! Короткое название ebuild "aterm" неоднозначно. Пожалуйста,
!!! вместо него укажите одно из полных названий ebuild:
    dev-libs/aterm
    x11-terms/aterm )
```

Название приложения, которое вы собираетесь установить, соответствует более чем одному пакету. Требуется также указать название категории. Portage предложит вам возможные варианты.

Циклические зависимости

Пример предупреждения Portage о циклических зависимостях:

```
!!! Error: circular dependencies:

ebuild / net-print/cups-1.1.15-r2 depends on ebuild /
app-text/ghostscript-7.05.3-r1
ebuild / app-text/ghostscript-7.05.3-r1 depends on ebuild /
net-print/cups-1.1.15-r2

( !!! Ошибка: циклические зависимости:
    ebuild / net-print/cups-1.1.15-r2 зависит от ebuild /
```

```
app-text/ghostscript-7.05.3-r1  
ebuild / app-text/ghostscript-7.05.3-r1 зависит от ebuild /  
net-print/cups-1.1.15-r2 )
```

Два или более пакета, которые вы хотите установить, взаимно зависимы, и в результате их установка невозможна. Скорее всего, это ошибка в дереве портежей. Пожалуйста, выждав время, обновите дерево портежей, и попробуйте снова. Вы можете также проверить, есть ли эта ошибка в [bugzilla](#), и если нет, сообщить о ней.

Ошибка извлечения

Пример предупреждения Portage об ошибке извлечения:

```
!!! Fetch failed for sys-libs/ncurses-5.4-r5, continuing...  
(...)  
!!! Some fetch errors were encountered. Please see above for details.  
( !!! Ошибка при извлечении sys-libs/ncurses-5.4-r5, продолжение...  
(...)  
!!! При извлечении произошли ошибки. Подробности выше. )
```

Portage не смогла загрузить исходный код данного приложения и попытается продолжить установку других приложений (если запланирована). Эта ошибка может произойти из-за неправильно синхронизированного зеркала, или из-за того, что ebuild указывает на неверное место. Сервер, где находятся исходные коды, также может почему-либо не работать.

Повторите действие через час, чтобы посмотреть, повторится ли эта ошибка.

Задача системного профиля

Пример предупреждения Portage о пакете, защищенном профилем:

```
!!! Trying to unmerge package(s) in system profile. 'sys-apps/portage'  
!!! This could be damaging to your system.  
( !!! Попытка удаления пакетов из системного профиля. 'sys-apps/portage'  
!!! Это может повредить вашей системе. )
```

Вы попросили удалить пакет, входящий в состав базовых пакетов вашей системы. Он отмечен в вашем профиле как обязательный, и его не следует удалять из системы.

USE-флаги

- [USE-флаги](#)
- [Что такое USE-флаги?](#)
- [Смысл USE-флагов](#)
- [Определение USE-флагов](#)
- [Какие USE-флаги существуют?](#)
- [Использование USE-флагов](#)
- [Объявление постоянных USE-флагов](#)
- [Объявление USE-флагов для отдельных пакетов](#)
- [Объявление временных USE-флагов](#)
- [Наследование](#)
- [Адаптация всей системы под новые USE-флаги](#)
- [USE-флаги отдельных пакетов](#)
- [Просмотр доступных USE-флагов](#)

Что такое USE-флаги?

Смысл USE-флагов

Устанавливая Calculate (или любой другой дистрибутив), вы выбираете те или иные возможности в зависимости от среды, с которой работаете. Установка сервера отличается от установки рабочей станции, а установка игровой станции - от платформы 3D-рендеринга.

Это касается не только того, какие пакеты устанавливать, но и какие функции определенных пакетов должны поддерживаться. Если вам не нужен OpenGL, то зачем вам его ставить и встраивать поддержку OpenGL в большинство программ? Если вы не собираетесь использовать KDE, зачем собирать пакеты с его поддержкой, если они работают и без этого?

USE флаги - простой способ описания рабочей среды, чтобы помочь пользователям в выборе того, что устанавливать/активировать, а что - нет. Это позволяет пользователю решить, что же ему на самом деле надо, и облегчить работу с Portage - системой управления пакетами.

Определение USE-флагов

Рассмотрим USE-флаги. USE-флаг - это ключевое слово, включающее сведения о поддержке и зависимостях определенного понятия или функции. При определении какого-либо USE-флага, Portage узнает, что вам нужна поддержка соответствующей функции. Конечно, это также влияет на сведения о зависимостях пакета.

Давайте рассмотрим конкретный пример - ключевое слово kde. Если в вашей переменной USE нет этого слова, то все пакеты, где поддержка KDE является необязательной, собираются без нее. Все пакеты, где зависимость от KDE является необязательной, устанавливаются без установки библиотек KDE (по зависимости). Если же вы определите ключевое слово kde, то эти пакеты будут собираться с поддержкой KDE, а KDE будет установлен в качестве необходимого.

Правильно определяя ключевые слова, вы создаете систему, подогнанную специально для ваших нужд.

Какие USE-флаги существуют?

Есть два типа USE-флагов: глобальные и локальные.

- *Глобальный* USE-флаг используется несколькими пакетами и является системным. Это то, что большинство видит в качестве USE-флагов.
- *Локальный* USE-флаг используется единичным пакетом для настройки определенных параметров самого пакета.

Список доступных глобальных USE-флагов можно найти [здесь](#) или локально в /usr/portage/profiles/use.desc.

Список локальных USE-флагов находится в вашей системе в /usr/portage/profiles/use.local.desc.

Использование USE-флагов

Объявление постоянных USE-флагов

Как сказано ранее, все USE-флаги объявляются в переменной USE. Каждый дистрибутив Calculate Linux имеет свой набор USE-флагов, оптимизированный под конкретные задачи.

Профиль, на который ориентируется ваша система, указывается символьной ссылкой /etc/make.profile. Каждый профиль основывается на предыдущем, более крупном, а итоговый складывается из всех профилей. Верхним является базовый профиль (/usr/portage/profiles/base).

В качестве примера можно посмотреть значение USE флагов профиля Calculate Directory Server:

```
... USE="acl amd64 bash-completion berkdb bittorrent bzip2 cli cracklib
crypt cups cxx dri exif foomaticdb fortran ftp gdbm gif gpm iconv imap
ipv6 jpeg jpeg2k ldap logrotate maildir mmx modules mudflap multilib
ncurses nfs nls nptl nptlonly openmp pam pcre perl png pppd python radius
readline samba session sse sse2 ssl sysfs tcpd tiff truetype unicode
userlocales xorg zlib" ...
```

Как видите, эта переменная уже содержит достаточно много ключевых слов.

Для изменения значения по умолчанию, нужно добавлять или удалять ключевые слова из переменной USE. Это делается глобально, определением переменной USE в </etc/make.conf>. В эту переменную можно добавить нужные вам USE-флаги, или удалить ненужные. Для удаления флага, его надо указывать со знаком минус в виде приставки («-»).

Например, чтобы убрать поддержку KDE и QT, но добавить поддержку ldap, можно определить в </etc/make.conf> переменную USE следующего вида:

```
USE="-kde -qt ldap"
```

Дистрибутивы Calculate Linux по умолчанию используют бинарный профиль, игнорирующий Ваши изменения USE флагов. Чтобы использовать изменения, во время установки или обновления пакетов используйте флаг -N (или --newuse)". Пример:

```
emerge -uN world
```

Объявление USE-флагов для отдельных пакетов

Иногда нужно определить некоторые USE-флаги только для одного или нескольких пакетов, не трогая системных настроек. Для этого необходимо создать файл в каталоге [/etc/portage/package.use/](/etc/portage/package.use) и отредактировать его значение. Имя файла может быть любым, удобным для вас.

Например, вам не нужна глобальная поддержка berkdb, но она необходима в mysql. Пример файла </etc/portage/package.use/mysql>:

```
dev-db/mysql berkdb
```

Естественно, можно в явном виде *отключить* USE-флаги для определенного пакета. Например, если вам не нужна поддержка java в PHP. Пример файла </etc/portage/package.use/nojava>:

```
dev-php/php -java
```

Объявление временных USE-флагов

Иногда необходимо установить какой-то USE-флаг только на один раз. Вместо того, чтобы дважды редактировать </etc/make.conf> (сначала добавить изменения USE, а потом удалить), можно просто объявить USE как переменную среды. Помните, что при переустановке или обновлении приложения (явном или в составе обновления системы) ваши изменения будут потеряны!

Например, уберем java из значения USE на время установки firefox:

```
USE="-java" emerge firefox
```

Наследование

Конечно же, существует определенная последовательность формирования значения USE. Вы же не хотите объявить USE="-java" только для того, чтобы узнать, что java все еще включена из-за значения с более высоким приоритетом. Последовательность установки значения USE в порядке приоритета (от меньшего к большему) такова:

- значение USE по умолчанию, объявленное в файлах make.defaults в составе вашего профиля
- значение, определенное пользователем в </etc/make.conf>
- значение, указанное пользователем в файлах [/etc/portage/package.use/](/etc/portage/package.use)

- значение, определенное пользователям в переменной среды

Чтобы узнать, какие же настройки USE в конечном счете видит Portage, запустите `emerge --info`. Эта команда выводит значения всех переменных (включая USE), используемые Portage:

```
emerge --info
```

Адаптация всей системы под новые USE-флаги

Если вы изменили свои USE-флаги и хотите обновить всю систему в соответствии с новым значением USE, запустите emerge с параметром --newuse:

Пример пересборки всей системы:

```
emerge -uDN world
```

Теперь запустите функцию Portage depclean, чтобы удалить условные зависимости, присутствующие в "старой" системе, но больше не нужные при новом составе USE-флагов.

Предупреждение: Запуск emerge -c (или --depclean) является опасной операцией, которую следует использовать с осторожностью. Дважды проверьте список «ненужных» пакетов и убедитесь, что не удалятся нужные пакеты. В следующем примере мы добавляем ключ -a, чтобы depclean потребовал подтверждения перед удалением.

Удаление ненужных пакетов:

```
emerge -ac
```

Когда depclean закончит свою работу, запустите revdep-rebuild, чтобы пересобрать программы, динамически связанные с библиотеками, входящими в потенциально удаленные пакеты:

```
revdep-rebuild
```

После выполнения всех этих действий, ваша система будет полностью использовать новые значения USE-флагов.

USE-флаги отдельных пакетов

Просмотр доступных USE-флагов

Возьмем, к примеру, firefox - какие USE-флаги он может использовать? Чтобы это выяснить, запустим emerge с параметрами -p (или --pretend) и -v (или --verbose). Пример:

```
# emerge -pv firefox
These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild N      ] www-client/firefox-3.6.13  USE="alsa bindist custom-optimization dbus ipc libnotify -gnome -java -startup-notification -system-sqlite -wifi" LINGUAS="bg de en es fr it pl pt_BR ru uk -af -ar -as -be -bn -bn_BD -bn_IN -ca -cs -cy -da -el -en_GB -en_US -eo -es_AR -es_CL -es_ES -es_MX -et -eu -fa -fi -fy -fy_NL -ga -ga_IE -gl -gu -gu_IN -he -hi -hi_IN -hr -hu -id -is -ja -ka -kk -kn -ko -ku -lt -lv -mk -ml -mr -nb -nb_NO -nl -nn -nn_NO -oc -or -pa -pa_IN -pt -pt_PT -rm -ro -si -sk -sl -sq -sr -sv -sv_SE -ta -ta_LK -te -th -tr -vi -zh_CN -zh_TW" 0 kB
```

emerge - не единственное средство для решения этой задачи. Существует программа equery, специально предназначенная для вывода информации о пакетах. Для просмотра USE-флагов какого-нибудь пакета запустим equery с аргументом uses. Пусть это будет пакет gnumeric:

```
# equery uses app-office/gnumeric-1.10.13 -a
[ Searching for packages matching app-office/gnumeric-1.10.13... ]
```

```
[ Colour Code : set unset ]
[ Legend : Left column (U) - USE flags from make.conf
[           : Right column (I) - USE flags packages was installed with ]
[ Found these USE variables for app-office/gnumeric-1.10.13 ]
U I
- - gnome   : Adds GNOME support
+ + perl    : Adds support/bindings for the Perl language
+ + python  : Adds support/bindings for the Python language
```

Возможности Portage

- [Возможности Portage](#)
- [Возможности Portage](#)
- [Распределенная компиляция](#)
- [Использование distcc](#)
- [Установка distcc](#)
- [Подключение поддержки Portage](#)
- [Кэширование компиляции](#)
- [О средстве ccache](#)
- [Установка ccache](#)
- [Подключение поддержки Portage](#)
- [Использование ccache для компиляции Си не в Portage](#)
- [Поддержка двоичных пакетов](#)
- [Создание готовых \(заранее собранных\) пакетов](#)
- [Установка двоичных пакетов](#)

Возможности Portage

В Portage есть несколько дополнительных возможностей. Многие из этих возможностей полагаются на определенные программы, повышающие производительность, надежность, безопасность и т.п.

Для включения и выключения определенных возможностей Portage нужно редактировать в файле `/etc/make.conf` переменную `FEATURES`, в которой перечислены ключевые слова, разделенные пробелами, обозначающие различные возможности. Иногда для использования соответствующих возможностей потребуется установка дополнительных утилит.

Здесь перечислены не все возможности, поддерживаемые Portage. Полный перечень представлен на странице справки </etc/make.conf>.

Чтобы узнать, какие возможности включены по умолчанию, запустите `emerge --info` и найдите переменную `FEATURES` (или отфильтруйте ее с помощью `grep`):

```
emerge --info | grep FEATURES
```

Распределенная компиляция

Использование distcc

distcc - программа, распределяющая компиляцию по нескольким, не обязательно одинаковым, машинам в сети. Клиент distcc посыпает всю необходимую информацию на доступные серверы distcc (на которых выполняется distccd), чтобы они могли компилировать для клиента части исходного кода. Чистый выигрыш - более быстрая компиляция.

Установка distcc

Distcc поставляется с графическим монитором (средством контроля), позволяющим отслеживать задачи, которые ваш компьютер отсылает на компиляцию. Если вы используете Gnome, тогда добавьте "gnome" к переменной USE. А если вы не пользуетесь Gnome, но при этом хотите пользоваться монитором, добавьте "gtk" к переменной USE.

Установка distcc:

```
emerge distcc
```

Подключение поддержки Portage

Добавьте distcc к переменной FEATURES в файле /etc/make.conf. Затем отредактируйте переменную MAKEOPTS, как вам нравится. Известная рекомендация - указывать директиву "-jX", где X - число центральных процессоров, на которых работает distccd (включая текущий компьютер) плюс один; у вас могут получиться лучшие результаты и с другими значениями.

Теперь запустите distcc-config и введите список доступных серверов distcc. Для простоты примера, предположим, что доступные серверы DistCC - 192.168.1.102 (текущий компьютер), 192.168.1.103 и 192.168.1.104 (два «удаленных» компьютера):

```
distcc-config --set-hosts "192.168.1.102 192.168.1.103 192.168.1.104"
```

Не забудьте также запустить демон distccd:

```
rc-update add distccd default  
/etc/init.d/distccd start
```

Кэширование компиляции

О средстве ccache

ccache - это быстрый кэш компилятора. Когда вы компилируете программу, он кэширует промежуточные результаты так, что всякий раз, когда вы перекомпилируете ту же самую программу, время компиляции значительно сокращается. В типичных случаях общее время компиляции может сокращаться в 5-10 раз.

Более подробно про ccache можно узнать на [сайте проекта](#).

Установка ccache

Для установки ccache, выполните emerge ccache:

```
emerge ccache
```

Подключение поддержки Portage

Откройте /etc/make.conf и добавьте ccache к переменной FEATURES. Затем добавьте новую переменную по имени CCACHE_SIZE (размер кэша), и установите её равной "2G":

```
CCACHE_SIZE="2G"
```

Для проверки работоспособности ccache, запросите статистику ccache.

```
ccache -s
```

Из-за того, что Portage использует другой домашний каталог ccache, вам также потребуется установить переменную CCACHE_DIR:

```
CCACHE_DIR="/var/tmp/ccache"
```

Домашний каталог ccache по умолчанию - `/var/tmp/ccache`; изменить это назначение можно, определив переменную `CCACHE_DIR` в `/etc/make.conf`.

Однако, при запуске ccache используется каталог по умолчанию, `${HOME} /.ccache`, вот почему при запросе статистики (Portage) ccache требуется определять переменную `CCACHE_DIR`.

Использование ccache для компиляции Си не в Portage

Если вы хотите использовать `ccache` для компиляций не в Portage, добавьте `/usr/lib/ccache/bin` в начало вашей переменной PATH (перед `/usr/bin`). Это можно сделать, отредактировав `/etc/env.d/00basic`, который является первым файлом среды, где определяется переменная PATH: `PATH="/usr/lib/ccache/bin:/opt/bin"`

Поддержка двоичных пакетов

Создание готовых (заранее собранных) пакетов

Portage поддерживает установку заранее собранных готовых пакетов. Несмотря на то, что для большинства дистрибутивов Calculate Linux есть поддержка заранее собранных пакетов, их количество ограничено составом пакетов дистрибутива. Вам также может понадобится подготовить пакеты с необходимыми вам [USE-флагами](#).

Чтобы создать двоичный пакет, можно использовать `quicpkg`, если пакет уже установлен в вашей системе, или `emerge` с параметрами `--buildpkg` или `--buildpkgonly`.

Если вы хотите, чтобы Portage создавал двоичные пакеты из каждого пакета, который вы будете устанавливать, в `/etc/make.conf` добавьте `buildpkg` к переменной FEATURES.

Установка двоичных пакетов

Дистрибутивы [Calculate Linux Desktop](#) (KDE и XFCE) и [Calculate Directory Server](#) используют бинарное хранилище обновлений содержащее пакеты, входящие в образ дистрибутива. Каждый из перечисленных дистрибутивов имеет как обычный профиль, так и бинарный, используемый по умолчанию. В бинарном профиле доступна только одна стабильная версия для каждого двоичного пакета.

Обратите внимание, перед установкой двоичного пакета из хранилища нужно выполнить обновление оверлея `Calculate layman -s calculate`.

Помимо маскировки, профиль устанавливает переменной FEATURES значение `getbinpkg`, отдавая приоритет установке бинарных пакетов.

Путь к хранилищу указан в переменной `PORTRAGE_BINHOST` в профиле дистрибутива. Для примера, в файле `/etc/make.conf` приведены пути к альтернативным хранилищам.

Вы можете устанавливать двоичные пакеты используя и обычный профиль. Для этого указывайте в команде `emerge` параметр `-g` (или `--getbinpkg`) вместе с параметром `-k` (или `--usepkg`). Первый указывает `emerge` загрузить двоичный пакет с сервера, определенного раньше, а второй сообщает `emerge`, что до загрузки исходных кодов и их компиляции сначала нужно попытаться установить этот двоичный пакет.

Например, чтобы установить `gnumeric` из двоичных пакетов:

```
emerge -kg gnumeric
```

Подробную информацию о параметрах установки двоичных пакетов можно найти на странице [справки emerge](#).

Переменные среды

- [Переменные среды](#)
- [Переменные среды](#)
- [Что это такое?](#)

- [Важные примеры](#)
- [Глобальное определение переменных](#)
- [Каталог /etc/env.d](#)
- [Сценарий env-update](#)
- [Локальное определение переменных](#)
- [Пользовательские переменные](#)
- [Сеансовые переменные](#)

Переменные среды

Что это такое?

Переменная среды --- это именованный объект, который содержит информацию, используемую одним или несколькими приложениями. Многие пользователи (особенно новички в Linux) находят этот подход несколько странным или неуправляемым. Но это впечатление ошибочно: используя переменные среды, можно очень легко изменить настройку разнообразных программ.

Важные примеры

В следующей таблице описывается ряд переменных, используемых в системе Linux. Примеры их значений приведены далее. **Переменная Описание**

PATH В этой переменной содержится список каталогов, разделенных двоеточиями, в которых система ищет исполняемые файлы. Если вы вводите имя исполняемого файла например ls, rc-update или emerge), который не находится ни в одной из перечисленных здесь каталогов, этот файл не запустится (если, конечно, вы не указали полный путь, например /bin/ls).

ROOTPATH У этой переменной такое же значение, что и у PATH, но в ней перечисляются только те каталоги, которые нужно просматривать при вводе команды пользователем с правами root.

LDPATH В этой переменной содержится список каталогов, разделенных двоеточиями, в которых динамический компоновщик ищет библиотеки.

MANPATH В этой переменной содержится список каталогов, разделенных двоеточиями, в которых команда man ищет страницы справки.

INFODIR В этой переменной содержится список каталогов, разделенных двоеточиями, в которых команда info ищет info-страницы.

PAGER В этой переменной содержится путь к программе, позволяющей постранично просматривать содержимое файлов, например less или more.

EDITOR В этой переменной содержится путь к программе, используемой для изменения файлов, например vi или nano.

KDEDIRS В этой переменной содержится список каталогов, разделенных двоеточиями, в которых находятся ресурсы KDE.

CLASSPATH В этой переменной содержится список каталогов, разделенных двоеточиями, в которых находятся классы Java.

CONFIG_PROTECT В этой переменной содержится список каталогов, защищаемых Portage при обновлении, разделенных пробелами.

CONFIG_PROTECT_MASK В этой переменной содержится список каталогов, исключаемых из защиты Portage при обновлении, разделенных пробелами.

Ниже представлен пример определения всех этих переменных:

```
PATH="/bin:/usr/bin:/usr/local/bin:/opt/bin:/usr/games/bin"
ROOTPATH="/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/sbin:/usr/local/bin"
LDPATH="/lib:/usr/lib:/usr/local/lib:/usr/lib/gcc-lib/i686-pc-linux-gnu/3.2.3"
MANPATH="/usr/share/man:/usr/local/share/man"
```

```
INFODIR="/usr/share/info:/usr/local/share/info"
PAGER="/usr/bin/less"
EDITOR="/usr/bin/vim"
KDEDIRS="/usr"
CLASSPATH="/opt/blackdown-jre-1.4.1/lib/rt.jar:."
CONFIG_PROTECT="/usr/X11R6/lib/X11/xkb /opt/tomcat/conf \
    /usr/kde/3.1/share/config
/usr/share/texmf/tex/generic/config/ \
    /usr/share/texmf/tex/plate/latex/config/ /usr/share/config"
CONFIG_PROTECT_MASK="/etc/gconf"
```

Глобальное определение переменных

Каталог /etc/env.d

Для того, чтобы определить эти переменные централизованно, в Calculate существует каталог `/etc/env.d`. В нём находится ряд файлов, например, `00basic`, `05gcc` и так далее, в которых определяются переменные, необходимые программам, указанным в названии файлов.

Например, при установке gcc ebuild создает файл `/etc/env.d/05gcc`, содержащий следующие определения переменных:

```
PATH="/usr/i686-pc-linux-gnu/gcc-bin/3.2"
ROOTPATH="/usr/i686-pc-linux-gnu/gcc-bin/3.2"
MANPATH="/usr/share/gcc-data/i686-pc-linux-gnu/3.2/man"
INFOPATH="/usr/share/gcc-data/i686-pc-linux-gnu/3.2/info"
CC="gcc"
CXX="g++"
LDPATH="/usr/lib/gcc-lib/i686-pc-linux-gnu/3.2.3"
```

В других дистрибутивах вам предлагается изменять или добавлять определения переменных среды в `/etc/profile` или где-нибудь еще. Calculate, с другой стороны, облегчает вам поддержку и управление переменными среды, избавляя от необходимости уделять внимание многочисленным файлам, содержащим определения переменных.

Например, когда обновляется gcc, также без малейшего участия пользователя обновляется и `/etc/env.d/05gcc`.

От этого выигрывает не только Portage, но и вы, пользователь. Иногда от вас может потребоваться глобальная установка какой-нибудь переменной. Возьмем, к примеру, переменную `http_proxy`. Вместо того, чтобы возиться с `/etc/profile`, теперь можно просто создать файл (`/etc/env.d/99local`) и добавить нужные определения туда:

```
http_proxy="proxy.server.com:8080"
```

Используя один и тот же файл для всех своих переменных, вы можете быстро увидеть все определенные вами переменные вместе.

Сценарий env-update

Переменная PATH определяется в нескольких файлах в `/etc/env.d`. Нет, нет это не ошибка: при запуске env-update различные определения объединяются перед обновлением переменных среды, позволяя пакетам (или пользователям) добавлять собственные значения переменных, не влияя на уже существующие.

Сценарий env-update объединяет значения переменных из файлов, находящихся в `/etc/env.d`, в алфавитном порядке. Имена файлов должны начинаться с двух десятичных цифр. Порядок обновления, используемый env-update:

```
00basic      99kde-env      99local
+-----+-----+-----+
PATH="/bin:/usr/bin:/usr/kde/3.2/bin:/usr/local/bin"
```

Объединение выполняется не всегда, а только для следующих переменных: KDEDIRS, PATH, CLASSPATH, LDPATH, MANPATH, INFODIR, INFOPATH, ROOTPATH, CONFIG_PROTECT, CONFIG_PROTECT_MASK, PRELINK_PATH и PRELINK_PATH_MASK. Для всех остальных переменных используется значение, определенное в последнем из файлов (по алфавиту в каталоге /etc/env.d).

При запуске сценария env-update создаются все переменные среды, и помещаются в /etc/profile.env (используемый файлом /etc/profile). Кроме того, на основе значения LDPATH создается /etc/ld.so.conf. После этого запускается ldconfig, чтобы вновь создать файла /etc/ld.so.cache, используемый динамическим компоновщиком.

Если вы хотите, чтобы результаты работы env-update вступили в силу немедленно, для обновления среды выполните следующую команду:

```
env-update && source /etc/profile
```

Примечание: Эта команда обновляет переменные только в текущем терминале, в новых консолях и их потомках. То есть, если вы работаете в X11, потребуется или набирать source /etc/profile в каждом открываемом терминале, или перезапустить X, чтобы все новые терминалы обращались к новым переменным. Если вы используете диспетчер входа в систему, станьте пользователем с правами root и наберите /etc/init.d/xdm restart. Если нет, вам придется выйти и снова войти в систему, чтобы X порождала потомков, использующих новые значения переменных.

Локальное определение переменных

Пользовательские переменные

Далеко не всегда нужно определять переменные глобально. Например, вам может понадобиться добавить /home/my_user/bin и текущий рабочий каталог (где вы находитесь) к переменной PATH, но при этом не нужно, чтобы это добавление появилось и в переменной PATH у всех остальных пользователей. Если вы хотите определить переменную среды локально, используйте ~/.bashrc или ~/.bash_profile. Пример расширения PATH в ~/.bashrc для локальных нужд:

```
#+(двоеточие без последующего указания каталога означает текущий рабочий каталог)
PATH="${PATH}:/home/my_user/bin:"
```

Обновление вашей переменной PATH произойдет, когда вы выйдете и снова войдете в систему.

Сессионные переменные

Иногда нужны еще более жесткие ограничения. Вам может потребоваться возможность запуска исполняемых файлов из специально созданного временного каталога без указания полного пути к ним, и без изменения файла ~/.bashrc ради нескольких минут.

В этом случае можно просто определить переменную PATH для текущего сеанса командой export. Переменной будет присвоено временное значение до тех пор, пока вы не завершите сеанс. Пример определения сессионной переменной среды:

```
export PATH="${PATH}:/home/my_user/tmp/usr/bin"
```

Файлы и каталоги

- [Файлы и каталоги](#)
- [Файлы Portage](#)

- [Директивы настройки](#)
- [Конфигурация, определяемая профилем](#)
- [Конфигурация, задаваемая пользователем](#)
- [Изменение файлов Portage и размещения каталогов](#)
- [Хранение файлов](#)
- [Дерево Portage](#)
- [Двоичные пакеты](#)
- [Исходные коды](#)
- [Файлы RPM](#)
- [База данных Portage](#)
- [Кэш Portage](#)
- [Сборка программного обеспечения](#)
- [Временные файлы Portage](#)
- [Каталог сборки](#)
- [Размещение «живой файловой системы»](#)
- [Ведение журнала](#)
- [Журнал Ebuild](#)

Файлы Portage

Директивы настройки

Настройки Portage по умолчанию хранятся в `/etc/make.globals`. Когда вы откроете этот файл, вы увидите, что все настройки представляют собой переменные. Что означает каждая из переменных, описано ниже.

Так как многие директивы отличаются в зависимости от используемой архитектуры, к Portage прилагаются настройки по умолчанию, которые входят в ваш профиль. На ваш профиль указывает символьская ссылка `/etc/make.profile`. Настройка Portage выполняется с помощью файлов `make.defaults` вашего профиля и всех родительских профилей. Более подробно о профилях и каталоге `/etc/make.profile` мы расскажем позже.

Если вы планируете вносить изменения в конфигурационные переменные, *не* изменяйте `/etc/make.globals` или `make.defaults`. Вместо этого пользуйтесь файлом [`/etc/make.conf`](#), который имеет приоритет перед вышеуказанными файлами. Вы также обнаружите файл `/etc/make.conf.example`. Как понятно из его названия, это просто пример - Portage не использует этот файл.

Переменные Portage также можно устанавливать как переменные среды, но мы не рекомендуем этого делать.

Конфигурация, определяемая профилем

Мы уже встречались с каталогом `/etc/make.profile`. На самом деле это не каталог, а символьская ссылка на профиль, по умолчанию на тот, что содержится в `/usr/portage/profiles`, однако вы можете создавать свои собственные профили где угодно и ссылаться на них. Профиль, указанный ссылкой, является профилем, к которому принадлежит ваша система.

В профиле содержатся сведения для Portage, специфичные для архитектуры, такие как список пакетов, принадлежащих соответствующей системе, список неработоспособных (или замаскированных) пакетов, и т.д.

Конфигурация, задаваемая пользователем

Если вам необходимо изменить поведение Portage относительно установки программного обеспечения, вам потребуется отредактировать файлы, находящиеся в `/etc/portage`. Мы настоятельно рекомендуем вам пользоваться файлами из `/etc/portage`, не следует настраивать поведение Portage через переменные среды.

Внутри `/etc/portage` доступны следующие пути:

- директория `package.mask`, в которой можно создать файлы с перечислением пакетов, которые Portage никогда не следует устанавливать;
- директория `package.unmask`, в которой можно создать файлы со списком пакетов, для которых вы хотите иметь возможность установки, даже если разработчики Gentoo отговаривают вас от этого;
- директория `package.keywords`, в которой можно создать файлы с перечислением пакетов, которые должны быть доступны для установки, несмотря на то, что они не подходят для вашей системы или архитектуры (пока);
- директория `package.use`, в которой можно создать файлы, где перечислены значения USE-флагов, которые необходимо указывать для конкретных пакетов, а не для всей системы.

Дополнительные сведения о каталоге `/etc/portage`, а также список всех файлов, которые там можно создавать, находятся на справочной странице Portage, см. `man portage`.

Изменение файлов Portage и размещения каталогов

Ранее упомянутые конфигурационные файлы нельзя хранить где угодно - Portage всегда ищет свои настроочные файлы в строго определенных местах. Однако Portage также использует множество каталогов для других целей: каталог для сборки, место для хранения исходных кодов, место для дерева Portage, и т.д.

Для этих целей существуют хорошо известные каталоги по умолчанию, положение которых можно изменить на свой вкус, внеся изменения в `/etc/make.conf`. Оставшаяся часть этой главы посвящена описанию того, какие специальные места Portage использует для своих целей, и как изменить их расположение в файловой системе.

Этот документ не претендует на статус справочника. Если вам необходим полный объем информации, пожалуйста, обратитесь к страницам справки по Portage и `make.conf` (`man portage` и `man make.conf`).

Хранение файлов

Дерево Portage

Дерево Portage размещается, по умолчанию, в `/usr/portage`. Это определяется значением переменной `PORTEXEC`. Когда вы храните дерево Portage где-либо в другом месте (изменив эту переменную), не забывайте соответственно изменить символьическую ссылку `/etc/make.profile`.

Если вы измените переменную `PORTEXEC`, вам может потребоваться изменить и следующие переменные: `PKGDIR`, `DISTDIR`, `RPMDIR`, так как они не замечают изменений `PORTEXEC`. Это связано с особенностями их обработки Portage.

Двоичные пакеты

Несмотря на то, что Portage по умолчанию не использует прекомпилированное программное обеспечение, для него предусмотрена очень мощная поддержка. Если вы укажете Portage работать с прекомпилированными пакетами, они будут разыскиваться в `/var/calculate/remote/packages`. Это расположение определяется переменной `PKGDIR`.

Исходные коды

Исходные коды приложений хранятся в `/var/calculate/remote/distfiles` по умолчанию. Это определяется переменной `DISTDIR`.

Файлы RPM

Несмотря на то, что Portage не может использовать RPM-файлы, есть возможность их создания командой ebuild (см. [Использование ebuild](#)). По умолчанию Portage хранит RPM файлы в каталоге /usr/portage/rpm, как определяется переменной RPMDIR.

База данных Portage

Portage хранит состояние вашей системы (какие пакеты установлены, какие файлы относятся к определенным пакетам и т. п.) в /var/db/pkg. *Не изменяйте эти файлы вручную!* Это может разрушить знание вашей системы Portage.

Кэш Portage

Кэш Portage (включая сведения о времени изменения, виртуальные пакеты, информацию дерева зависимостей и т. д.) хранится в /var/cache/edb. Это место действительно является кэшем: вы можете его очистить в любой момент, когда не запущены приложения, связанные с Portage.

Сборка программного обеспечения

Временные файлы Portage

По умолчанию Portage хранит временные файлы в /var/tmp. За это отвечает переменная PORTAGE_TMPDIR.

Если вы измените переменную PORTAGE_TMPDIR, вам может потребоваться изменить и переменную BUILD_PREFIX, так как она не замечает изменений PORTAGE_TMPDIR. Это связано с особенностями ее обработки Portage.

Каталог сборки

Portage создает специфичные каталоги сборки для каждого пакета внутри /var/tmp/portage. Это расположение задается переменной BUILD_PREFIX.

Размещение «живой файловой системы»

По умолчанию Portage устанавливает все файлы в текущую файловую систему (/), но это можно изменить, установив переменную окружения ROOT. Это может оказаться полезным при построении новых образов системы.

Ведение журнала

Журнал Ebuild

Portage может создавать отдельные файлы журнала для каждого файла ebuild, но только тогда, когда переменная PORT_LOGDIR указывает на место, доступное для записи для Portage (пользователя portage). По умолчанию эта переменная не установлена.

Настройка с помощью переменных

- [Настройка с помощью переменных](#)
- [Настройка Portage](#)
- [Параметры сборки](#)
- [Параметры конфигурирования и компиляции](#)
- [Параметры установки](#)
- [Зашита конфигурационных файлов](#)
- [Места, защищаемые Portage](#)
- [Исключение каталогов](#)
- [Параметры скачивания](#)
- [Расположение сервера](#)

- [Команды для извлечения](#)
- [Настройки rsync](#)
- [Настройка Gentoo](#)
- [Выбор ветви](#)
- [Возможности Portage](#)
- [Поведение Portage](#)
- [Распределение ресурсов](#)
- [Настройки вывода](#)

Настройка Portage

Как отмечалось ранее, Portage настраивается с помощью множества переменных, которые задаются в файле [/etc/make.conf](#).

Параметры сборки

Параметры конфигурирования и компиляции

Когда Portage собирает приложения, компилятору и сценарию конфигурации передаются значения следующих переменных:

- `CFLAGS` и `CXXFLAGS` определяют желаемые флаги компилятора для C и C++
- `CHOST` определяет информацию об используемой платформе для сценария конфигурации приложения
- `MAKEOPTS` передается команде `make` и обычно применяется для установки степени распараллеливания компиляции. Более подробная информация о параметрах команды `make` находится на странице справки по `make`.

Переменная `USE` также используется при конфигуратории и компиляции, но о ней уже много и подробно говорилось в предыдущих главах.

Параметры установки

Когда Portage устанавливает (`merge`) новую версию программного продукта, файлы более старых версий удаляются из системы. Portage дает пользователю 5-ти секундную задержку перед стиранием старых версий. Эти 5 секунд задаются переменной `CLEAN_DELAY`.

Задача конфигурационных файлов

Места, защищаемые Portage

Portage записывает файлы, предоставляемые новой версией программы, поверх старых, если только эти файлы не расположены в защищенном месте. Защищенные каталоги определяются переменной `CONFIG_PROTECT`. Обычно, это места расположения файлов конфигурации. Каталоги в списке разделяются пробелами.

Файл, который должен быть записан в такой защищенный каталог, переименовывается, а пользователь получает предупреждение о наличии новой версии (обычно) файла конфигурации.

Узнать текущее значение `CONFIG_PROTECT` можно из сообщений `emerge --info`:

```
emerge --info | grep 'CONFIG_PROTECT='
```

Более подробная информация о защите конфигурационных файлов, осуществляемой системой Portage, доступна по команде `emerge`:

```
emerge --help config
```

Исключение каталогов

Чтобы снять защиту с определенных подкаталогов защищенного каталога, можно использовать переменную `CONFIG_PROTECT_MASK`.

Параметры скачивания

Расположение сервера

Если запрошенная информация или данные отсутствуют в вашей системе, Portage обращается за ними в интернет. Расположение серверов для различных каналов получения информации задается следующими переменными:

- `GENTOO_MIRRORS` определяет список адресов серверов, содержащих исходный код (`distfiles`)
- `PORTRAGE_BINHOST` указывает расположение определенного сервера, содержащего двоичные пакеты (`prebuilt packages`) для вашей системы

Третья переменная содержит расположение сервера `rsync`, который используется при обновлении вашего дерева портажей:

- `SYNC` указывает сервер, с которого Portage извлекает дерево портажей

Переменные `GENTOO_MIRRORS` и `SYNC` можно установить автоматически программой `mirrorselect`. Перед тем, как использовать, ее нужно установить, выполнив `emerge mirrorselect`. За дополнительной информацией обращайтесь к оперативной справке `mirrorselect`:

```
mirrorselect --help
```

Если вы вынуждены использовать прокси-сервер, для его указания можно использовать переменные `HTTP_PROXY`, `FTP_PROXY` и `RSYNC_PROXY`.

Команды для извлечения

Когда Portage требуется извлечь исходный код, по умолчанию используется `wget`. Вы можете это изменить с помощью переменной `FETCHCOMMAND`.

Portage может возобновлять скачивание частично загруженного исходного кода. По умолчанию используется `wget`, но это можно переопределить переменной `RESUMECOMMAND`.

Удостоверьтесь, что ваши команды `FETCHCOMMAND` и `RESUMECOMMAND` сохраняют исходный код в нужном месте. Внутри этих переменных следует использовать `\${URI}` и `\${DISTDIR}`, для указания расположения исходных кодов и `distfiles`, соответственно.

Также существует возможность определить индивидуальные настройки для различных протоколов, используя `FETCHCOMMAND_HTTP`, `FETCHCOMMAND_FTP`, `RESUMECOMMAND_HTTP`, `RESUMECOMMAND_FTP` и т.п.

Настройки `rsync`

Вы не можете заменить команду `rsync`, которую Portage использует для обновления дерева портажей, но можно установить несколько переменных, определяющих ее поведение:

- `RSYNC_EXCLUDEFROM` указывает на файл, где перечислены пакеты и/или категории, которые `rsync` должна игнорировать во время обновления.
- `RSYNC_RETRIES` определяет, сколько раз `rsync` должна пытаться соединиться с зеркалом, на которое указывает переменная `SYNC`. По умолчанию равна 3.
- `RSYNC_TIMEOUT` определяет количество секунд, в течение которого `rsync` соединение может бездействовать, перед тем как `rsync` сочтет его превысившим время ожидания. По умолчанию равна 180, но если вы используете соединение по модему или у вас медленный компьютер, возможно, следует установить значение этой переменной равным 300 или большим.

Настройка Gentoo

Выбор ветви

Используемую ветвь можно изменить переменной ACCEPT_KEYWORDS. По умолчанию используется стабильная ветвь для вашей архитектуры. Дополнительная информация о ветвях Gentoo находится в следующей главе.

Возможности Portage

Вы можете включить отдельные функции Portage с помощью переменной FEATURES. Возможности Portage рассматривались в предыдущих главах, например [Возможности Portage](#).

Поведение Portage

Распределение ресурсов

С помощью переменной PORTAGE_NICENESS можно увеличивать или уменьшать значение nice, с которым выполняется Portage. Значение PORTAGE_NICENESS прибавляется к текущему значению nice.

Более подробно о значениях nice написано в странице справки:

`man nice`

Настройки вывода

Переменная NOCOLOR (по умолчанию "false") определяет, следует ли Portage отключить цветовую раскраску своих сообщений.

Смешение ветвей программного обеспечения

- [Смешение ветвей программного обеспечения](#)
- [Использование одной ветви](#)
- [Стабильная ветвь](#)
- [Тестовая ветвь](#)
- [Одновременное использование стабильной и тестовой ветвей](#)
- [Местоположение package.keywords](#)
- [Тестирование определенных версий](#)
- [Использование заблокированных пакетов](#)
- [Расположение package.unmask](#)
- [Местоположение package.mask](#)

Использование одной ветви

Стабильная ветвь

Переменная ACCEPT_KEYWORDS определяет, какую из ветвей использовать в вашей системе. По умолчанию используется стабильная ветвь для вашей архитектуры, например x86.

Тестовая ветвь

Если вы желаете использовать наиболее свежее ПО, подумайте над использованием тестовой ветви. Чтобы Portage начала использовать тестовую ветвь, добавьте «~» перед названием вашей архитектуры.

Тестовая ветвь полностью соответствует своему названию: для тестирования. Если пакет находится в стадии тестирования, это означает, что разработчики считают, что пакет работоспособен, но тщательно он не протестирован. Вы можете оказаться первым, кто столкнется с какой-либо ошибкой. В этом случае вы можете создать [отчет об ошибке](#), чтобы разработчики узнали о ней.

Однако будьте готовы к тому, что могут возникнуть проблемы со стабильностью, неудовлетворительной поддержкой пакетов (например неправильные/отсутствующие зависимости), слишком частыми обновлениями (а в результате --- частыми сборками) или невозможностью собрать пакет. Если вы не знаете, как работает система и как разрешать возникающие проблемы, мы рекомендуем не отходить от стабильной и оттестированной ветви.

К примеру, для выбора тестовой ветви на архитектуре x86, отредактируйте [/etc/make.conf](#) и укажите в нем:

```
ACCEPT_KEYWORDS=~x86"
```

Если вы запустите обновление системы, то увидите, что многие пакеты нуждаются в обновлении. Обратите внимание, что после перехода на тестовую ветвь и обновления системы, как правило, нет простого пути назад к стабильной официальной ветви (конечно, кроме использования резервной копии).

Одновременное использование стабильной и тестовой ветвей

Местоположение package.keywords

Вы можете указать, чтобы Portage использовала тестовую ветвь только для определенных пакетов, а для остальной системы --- стабильную ветвь. Для этого добавьте категорию и имя пакета, для которого вы желаете использовать тестовую ветвь. Создайте файл с любым именем в директории `/etc/portage/package.keywords/` или отредактируйте `/etc/portage/package.keywords/custom`. Например, для использования тестовой ветви для gnumeric:

```
app-office/gnumeric ~x86
```

Тестирование определенных версий

Если вы желаете использовать конкретную версию ПО из тестовой ветви, но не хотите, чтобы Portage использовала тестовую ветвь для последующих версий этого ПО, можно указать в местоположении `package.keywords` номер необходимой версии. В этом случае вы обязаны использовать оператор `=`. Также можно указать диапазон версий, используя операторы

В любом случае, добавляя информацию о версии, вы должны использовать один из этих операторов. Если вы не указываете версию, эти операторы использовать нельзя.

В следующем примере мы просим Portage разрешить установку gnumeric-1.2.13:

```
=app-office/gnumeric-1.2.13 ~x86
```

Использование заблокированных пакетов

Расположение package.unmask

Если использование пакета было заблокировано, но вы желаете его использовать несмотря на причины блокировки, добавьте для него точно такую же строку, создав файл внутри каталога `/etc/portage/package.unmask`.

Например, если `=net-mail/hotwayd-0.8` заблокирован, то разблокировать его можно, прописав в `package.unmask` точно такую же строчку:

```
=net-mail/hotwayd-0.8
```

Местоположение package.mask

Если вы не хотите, чтобы Portage использовала какое-то конкретное ПО или конкретные версии ПО, вы можете его самостоятельно заблокировать, добавив соответствующую запись, создав файл внутри каталога `/etc/portage/package.mask`.

Если, к примеру, вы не хотите, чтобы Portage устанавливала исходные коды ядра новее, чем calculate-sources-2.6.36.3, добавьте такую строку в местоположение package.mask:

```
>sys-kernel/calculate-sources-2.6.36.3
```

Дополнительные средства Portage

- [Дополнительные средства Portage](#)
- [etc-update](#)
- [dispatch-conf](#)
- [quickpkg](#)

etc-update

etc-update --- это утилита, предназначенная для обновления в системе файлов `._cfg0000_<имя>`. Она обеспечивает интерактивную настройку установки и может также автоматически устанавливать тривиальные изменения. Файлы создаются `._cfg0000_<имя>` Portage, когда нужно заменить файл в каталоге, защищенным переменной CONFIG_PROTECT.

Выполнить etc-update довольно просто:

```
etc-update
```

После выполнения тривиальных обновлений, вы увидите запрос со списком защищенных файлов, ожидающих обновления. Внизу вам предложат следующие варианты:

```
Please select a file to edit by entering the corresponding number.  
(-1 to exit) (-3 to auto merge all remaining files)  
(-5 to auto-merge AND not use 'mv -i'): )
```

(Пожалуйста, выберите файл для правки, введя соответствующее число.
(-1 - выход) (-3 - автоустановка всех оставшихся файлов)
(-5 для автоустановки БЕЗ использования 'mv -i'):)

При вводе -1, etc-update выходит, прекращая последующие изменения. Если вы введете -3 или -5, все перечисленные файлы конфигурации заменяются более новыми версиями. Следовательно, очень важно сначала отобрать файлы, которые не следует автоматически обновлять. Для этого надо только вводить номер, указанный слева от файлов.

Например, выбираем файл конфигурации `/etc/pear.conf`:

```
Beginning of differences between /etc/pear.conf and  
/etc/._cfg0000_pear.conf
```

```
...  
End of differences between /etc/pear.conf and /etc/._cfg0000_pear.conf  
1) Replace original with update  
2) Delete update, keeping original as is  
3) Interactively merge original with update  
4) Show differences again
```

Теперь можно увидеть различия между двумя файлами. Если вы считаете, что обновленный файл конфигурации можно использовать без проблем, введите 1. Если вы считаете, что обновленный файл конфигурации не нужен, или не содержит новую или полезную информацию, введите 2. Если вы хотите обновить текущий файл в интерактивном режиме, введите 3.

Нет никакого смысла в подробном описании интерактивного обновления. Для полноты изложения, мы перечислим возможные команды, которые можно использовать при интерактивном слиянии двух файлов. Вас встречают две строки (одна исходная, вторая измененная) и запрос, в ответ на который можно ввести одну из следующих команд:

```
ed:      редактировать и использовать оба варианта, каждый пометить заголовком
eb:      редактировать и использовать оба варианта
el:      редактировать и использовать левый вариант
er:      редактировать и использовать правый вариант
e:       редактировать новую версию
l:       использовать левую версию
r:       использовать правую версию
s:       молча включить общие строки
v:       включить общие строки, сообщив подробности
q:       выход
```

Завершив обновление важных файлов конфигурации, вы можете автоматически обновить оставшиеся файлы конфигурации. etc-update выйдет, если не найдет других файлов, подлежащих обновлению.

dispatch-conf

С помощью dispatch-conf можно обновлять файлы конфигурации, сохраняя при этом историю изменений. dispatch-conf хранит различия между файлами конфигурации в виде заплаток или в системе управления версиями RCS.

Как и с etc-update, вы можете попросить сохранить файл конфигурации как есть, использовать новый файл конфигурации, редактировать текущий или объединить изменения интерактивно. Однако, у dispatch-conf также есть приятные дополнительные возможности:

- автоматическое обновление файлов, в которых обновились только комментарии
- автоматическое обновление файлов, которые отличаются только количеством пробелов

Убедитесь, что вы сначала отредактировали `/etc/dispatch-conf.conf` и создали каталог, прописанный в `archive-dir`.

За дополнительными сведениями обращайтесь к странице справки `dispatch-conf`:

```
man dispatch-conf
```

quickpkg

С quickpkg вы можете создавать архивы пакетов, уже установленных в системе. Эти архивы можно использовать в качестве двоичных пакетов. Запуск quickpkg прост: только укажите имена пакетов, которые нужно заархивировать.

Например, чтобы поместить в архив curl, arts и procps:

```
quickpkg curl arts procps
```

Двоичные пакеты будут храниться в `$PKGDIR/All` (по умолчанию --- `/var/calculate/remote/packages`). Символьные ссылки, указывающие на эти пакеты, помещаются в `$PKGDIR/<категория>`.

Отступление от официального дерева

- [Отступление от официального дерева](#)
- [Использование собственного дерева Portage](#)
- [Исключение пакета/категории](#)
- [Добавление неофициального сборочного файла ebuild](#)

- [Определение оверлейного каталога портежей](#)
- [Работа с несколькими оверлейными каталогами](#)
- [Программы, поддерживаемые не Portage](#)
- [Использование Portage с пакетами самостоятельной сборки](#)

Использование собственного дерева Portage

Исключение пакета/категории

Вы можете выборочно обновлять определенные категории/пакеты, игнорируя обновление других категорий/пакетов. Это достигается путем исключения таких категорий/пакетов программой rsync на этапе выполнения emerge --sync.

Вам потребуется определить имя файла, содержащего шаблоны исключаемых пакетов, в переменной RSYNC_EXCLUDEFROM в своем файле [/etc/make.conf](#):

```
RSYNC_EXCLUDEFROM=/etc/portage/rsync_excludes
```

Для примера исключим все игры в файле /etc/portage/rsync_excludes:

```
games - /*
```

Заметьте, однако, что это может привести к проблемам с зависимостями, так как новые разрешенные пакеты могут зависеть от других новых, но исключенных из обновления пакетов.

Добавление неофициального сборочного файла ebuild

Определение оверлейного каталога портежей

Вы можете указать Portage использовать сборочные файлы, не входящие в официальное дерево Portage. Создайте новый каталог (к примеру, /usr/local/portage), в котором будут находиться файлы ebuild сторонних разработчиков. Используйте в точности такую же структуру каталогов, как и в официальном дереве портежей!

Затем определите переменную PORTDIR_OVERLAY в /etc/make.conf, чтобы она указывала на ранее созданный каталог. Теперь при использовании Portage, эти сборочные файлы будут рассматриваться как часть системы, и не будут удаляться/перезаписываться при последующих запусках emerge --sync.

Работа с несколькими оверлейными каталогами

Для продвинутых пользователей, ведущих разработку в нескольких оверлейных каталогах, тестирующих пакеты перед включением в основное дерево портежей или просто желающих использовать неофициальные сборочные файлы ebuild из разных источников, в пакете app-portage/layman есть утилита layman, которая поможет поддерживать ваши оверлейные репозитории в актуальном состоянии.

Настройте layman, а затем добавьте интересующие вас оверлеи командой layman -a название\оверлея_

Предположим, что у вас есть два дополнительных репозитория с названиями java (для сборочных файлов разработок, ведущихся на java) и entapps (для внутренних приложений, разработанных на вашем предприятии). Вы можете обновить эти репозитории следующей командой:

```
layman -S
```

Программы, поддерживаемые не Portage

Использование Portage с пакетами самостоятельной сборки

Иногда вам может потребоваться сконфигурировать, установить и поддерживать программное обеспечение самостоятельно, без автоматизации со стороны Portage, не смотря на то, что оно поддерживается Portage. Наиболее известные случаи --- это исходные коды ядра и драйверы от nVidia. Вы можете настроить Portage так, чтобы системе стало известно, что определенные пакеты установлены вручную. Этот процесс называется *внедрение*, и поддерживается Portage посредством файла /etc/portage/profile/package.provided.

Например, если вы захотите сообщить Portage, что пакет vanilla-sources-2.6.37.2 установлен вручную, нужно добавить следующую строку в /etc/portage/profile/package.provided:

```
sys-kernel/vanilla-sources-2.6.37.2
```

Использование ebuild

- [Использование ebuild](#)
- [emerge и ebuild](#)
- [Ручная установка программ](#)
- [Извлечение исходных кодов и проверка контрольных сумм](#)
- [Распаковка исходных кодов](#)
- [Компиляция исходных кодов](#)
- [Установка файлов во временное место](#)
- [Помещение файлов в рабочую файловую систему](#)
- [Очистка временного каталога](#)
- [Дополнительные возможности Ebuild](#)
- [Запуск всех команд установки](#)
- [Выполнение действий по настройке](#)
- [Сборка пакета \(RPM\)](#)
- [Дополнительная информация](#)

emerge и ebuild

Программа ebuild --- это низкоуровневый интерфейс системы Portage. С ее помощью можно выполнять определенные действия над заданными сборками ebuild. Например, вы можете самостоятельно выполнить отдельные этапы установки.

Программа ebuild предназначена в основном для разработчиков, поэтому более подробная информация находится в [настольной книге разработчика](#) (англ.). Однако, мы расскажем, какие экземпляры ebuild вызываются системой Portage на разных этапах установки, и как выполнить пост-конфигурационные шаги, которые допускаются некоторыми пакетами.

Ручная установка программ

Извлечение исходных кодов и проверка контрольных сумм

Каждый раз, когда вы вызываете ebuild для какого-то ebuild-файла, проверяется совпадение контрольной суммы всех задействованных файлов с указаной в файлах Manifest или files/digest-<имя>-<версия>. Проверка выполняется после загрузки исходных кодов.

Чтобы загрузить исходные коды с помощью ebuild, запустите:

```
ebuild путь/к/файлу-ebuild fetch
```

Если контрольная сумма md5 сборочного файла не совпадает с той, что указана в файле `Manifest`, или же один из загруженных файлов не совпадает с описанием в файле `files/digest<пакет>`, вы получите сообщение об ошибке, похожее на такое:

```
!!! File is corrupt or incomplete. (Digests do not match)
>>> our recorded digest: db20421ce35e8e54346e3ef19e60e4ee
>>> your file's digest: f10392b7c0b2bbc463ad09642606a7d6
```

(!!! Файл поврежден или усечен. (Контрольные суммы не совпадают))

На следующей строке указывается проблемный файл.

Если вы абсолютно уверены, что загруженные исходные коды и сам сборочный файл `ebuild` именно те, что вам нужны, можете пересоздать файлы `Manifest` и `digest-<пакете>`, используя функцию `digest` программы `ebuild`. Пример создания новых файлов `Manifest` и `digest`:

```
ebuild путь/к/файлу-ebuild digest
```

Распаковка исходных кодов

Чтобы разпаковать исходные коды в `/var/tmp/portage` (или любой другой каталог, указанный в [`/etc/make.conf`](#)), запустите функцию `unpack` программы `ebuild`. Пример распаковки исходных кодов:

```
ebuild путь/к/файлу-ebuild unpack
```

Эта команда выполнит функцию `src_unpack()` программы `ebuild` (которая по умолчанию просто выполняет распаковку, если функция `src_unpack()` не определена). Все необходимые заплатки накладываются также на этом этапе.

Компиляция исходных кодов

Следующий шаг в процессе установки --- компиляция исходных кодов. Для этого выполняется функция `src_compile()` вашего сборочного файла. Если нужно, заодно выполняется конфигурация. Пример компиляции исходных кодов:

```
ebuild путь/к/файлу-ebuild compile
```

Если вы хотите изменить инструкции компиляции, советуем отредактировать функцию `src_compile()`. Однако, вы можете также обмануть Portage, заставив ее поверить, что программа `ebuild` уже завершила компиляцию. Запустите нужные команды самостоятельно и создайте пустой файл `.compile` в рабочем каталоге:

```
touch .compiled
```

Установка файлов во временное место

Следующий шаг --- установка всех необходимых файлов во временный каталог. В него помещаются все файлы, подлежащие включению в рабочую файловую систему. Вы можете выполнить этот этап, запустив функцию установки программы `ebuild`, которая выполняет функцию `src_install()` сборочного файла:

```
ebuild путь/к/файлу-ebuild install
```

Помещение файлов в рабочую файловую систему

Последний этап --- перенос всех файлов в рабочую файловую систему и их регистрация в системе Portage. В `ebuild` этот этап называется "qmerge", и включает следующие действия:

- выполняется функция `pkg_preinst()`, если она определена
- все файлы копируются в рабочую файловую систему
- файлы регистрируются в системе Portage

- выполняется функция `pkg_postinst()`, если она определена

Запустите функцию `qmerge` программы `ebuild`, чтобы выполнить этот этап:

```
ebuild путь/к/файлу-ebuild qmerge
```

Очистка временного каталога

Наконец, можно очистить временный каталог, используя команду `clean` программы `ebuild`:

```
ebuild путь/к/файлу-ebuild clean
```

Дополнительные возможности Ebuild

Запуск всех команд установки

С помощью функции `merge` программы `ebuild`, можно запустить команды извлечения, распаковки, компиляции, установки и помещения за один раз:

```
ebuild путь/к/файлу-ebuild merge
```

Выполнение действий по настройке

В некоторых приложениях содержатся инструкции по дальнейшей настройке установленного пакета. Эти инструкции могут потребовать участия пользователя, и, следовательно, не выполняться автоматически. Для запуска шагов настройки, указанных в необязательной функции `config()` сборочного файла, используйте команду `config` программы `ebuild`. Пример настройки пакета:

```
ebuild путь/к/файлу-ebuild config
```

Сборка пакета (RPM)

Вы можете попросить Portage создать двоичный пакет или даже RPM из вашего сборочного файла, воспользовавшись командами `package` и `rpm`, соответственно. Эти команды несколько различаются:

- команда `package` во многом похожа на `merge`, выполняя все необходимые шаги (извлечение, распаковку, компиляцию, установку) перед созданием пакета
- команда `rpm` собирает пакет RPM из файлов созданных после запуска окончания функции `install` программы `ebuild`

Пример создания пакетов:

```
 #(создание двоичного пакета, совместимого с Portage)
ebuild путь/к/файлу-ebuild package
```

```
 #(создание пакета RPM)
ebuild путь/к/файлу-ebuild rpm
```

Созданный RPM, однако, не будет содержать информацию о зависимостях из сборочного файла `ebuild`.

Дополнительная информация

За дополнительными сведениями о системе Portage, программе `ebuild` и сценариях `ebuild` обращайтесь к следующим страницам справки `man`:

```
man portage      (сама система Portage)
man emerge       (команда emerge)
man ebuild       (команда ebuild)
man 5 ebuild     (синтаксис файлов ebuild)
```

Кроме того, дополнительные сведения, относящиеся к разработке, находятся в [настольной книге разработчика](#) (англ.).

4. Утилиты Calculate

1. [Графический клиент утилит Calculate](#)
2. [Консольный клиент утилит Calculate](#)
3. [Сервер утилит Calculate](#)
4. [Шаблоны утилит Calculate](#)
5. [Переменные шаблонов](#)
6. [Хранение настроек профиля пользователя](#)
7. [Обновление системы cl-update](#)
8. [Смена профиля системы cl-update-profile](#)
9. [Сборка системы](#)
10. [Calculate API](#)

Графический клиент утилит Calculate

11. [Графический клиент утилит Calculate](#)
12. [Введение](#)
13. [Создание сертификата](#)
14. [Режимы запуска](#)
15. [Соединение и работа с сервером утилит](#)
16. [Панель меню](#)
17. [Возврат в Главное меню](#)
18. [Запущенные процессы](#)
19. [Информация о сессии](#)
20. [Настройка программы](#)
21. [Помощь](#)
22. [Выполнение методов на сервере утилит](#)
23. [Безопасность и разграничение прав](#)
24. [Запросы](#)
25. [Сертификаты](#)
26. [Права групп](#)

Введение

Клиент утилит Calculate служит для сетевого доступа по протоколу <https> к функциям сервера утилит Calculate 3. Установить клиент можно при помощи пакета `sys-apps/calculate-console-gui`.

Для запуска клиента выполните команду:

```
cl-console-gui
```

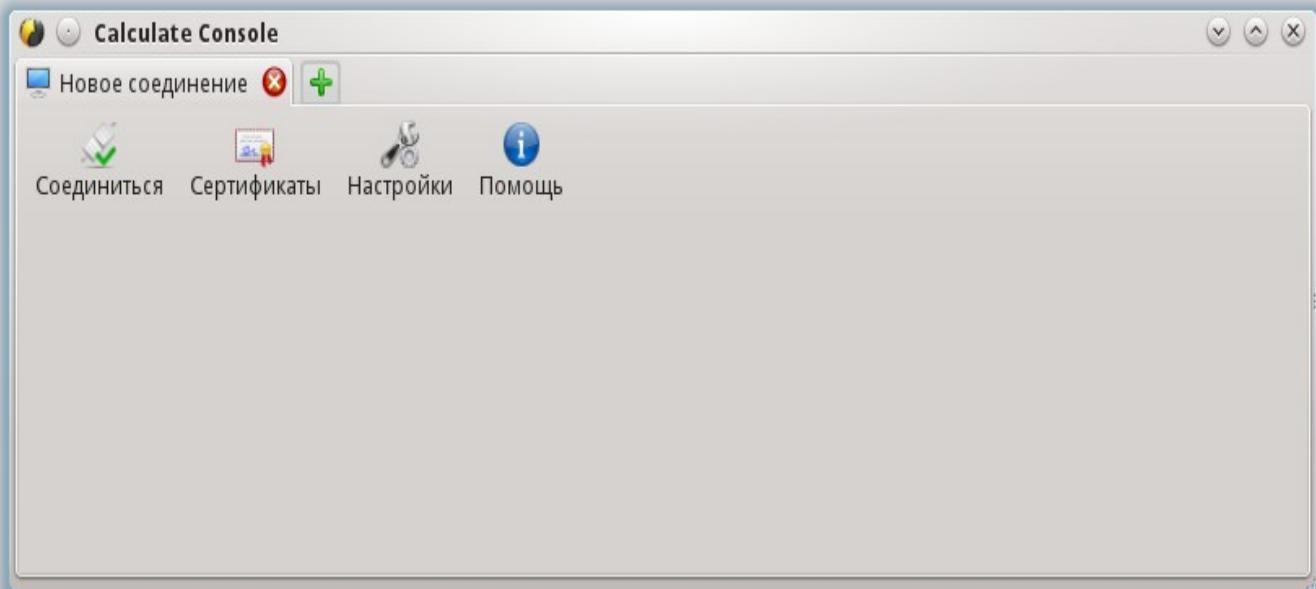


Рис. 1 Внешний вид окна cl-console-gui

Создание сертификата

Для работы с сервером утилит необходим под подписанный им сертификат. Для генерации запроса на подписание сертификата (далее запрос) и получения сертификата от сервера утилит используйте пункт "Сертификаты" в меню.

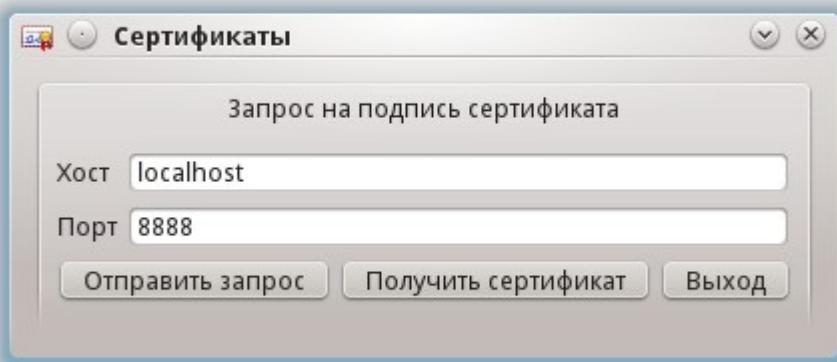


Рис. 2 Запрос на подпись сертификата

В поле *Хост* введите адрес хоста сервера утилит, в поле *Порт* номер порта, который прослушивает сервер (по умолчанию 8888). Используйте кнопку "Отправить запрос" для создания и отправления запроса на сервер утилит и кнопку "Получить сертификат" для получения подписанного сертификата с сервера утилит.

Для создания запроса и секретного ключа необходимо ввести некоторые данные в окне "Создание запроса на подпись сертификата".

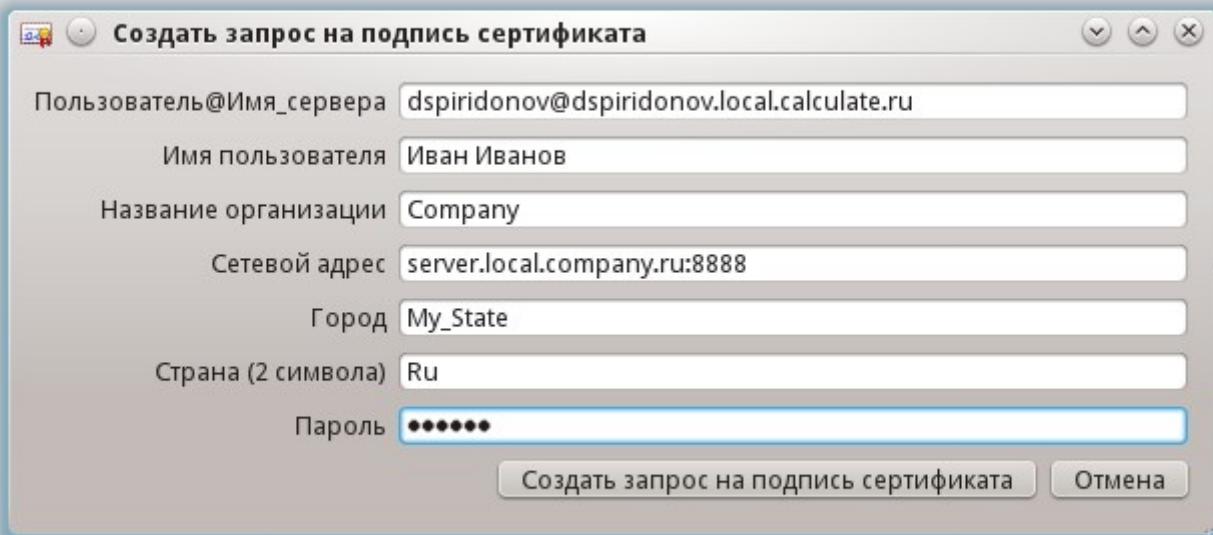


Рис. 3 Создания запроса на подпись сертификата

Большая часть данных, кроме названия организации, города и пароля, будет заполнена автоматически. Не меняйте автоматически подставленные значения, если не уверены, что значения верны. Используйте поле "Пароль", если необходимо дополнительное шифрование закрытого ключа. Введённый пароль будет запрашиваться для установки соединения с сервером утилиты.

После создания, запрос автоматически отправится на сервер утилиты и появится окно с номером запроса.

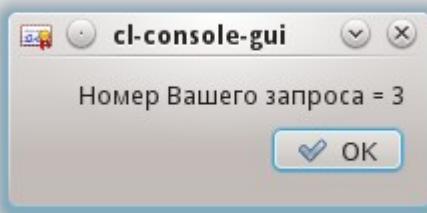


Рис. 4 Информация о номере запроса

Подписать запрос можно на сервере утилиты через клиента или с помощью команды:

```
cl-core --sign-client n
```

Где n - номер запроса.

После подписания сертификата на сервере, его необходимо забрать нажав на кнопке "Получить сертификат" в окне сертификата. Если запрос ещё не подписан или отвергнут, то появится соответствующее сообщение.

Режимы запуска

Графический клиент может быть запущен в одном из двух режимов:

- запуск в обычном режиме командой

```
cl-console-gui
```

- запуск отдельного метода с помощью ключа --method

```
cl-console-gui --method methodname
```

Пример запуска установки системы:

```
cl-console-gui --method install
```

Соединение и работа с сервером утилит

Для соединения с сервером утилит используйте пункт "Соединиться" в меню.

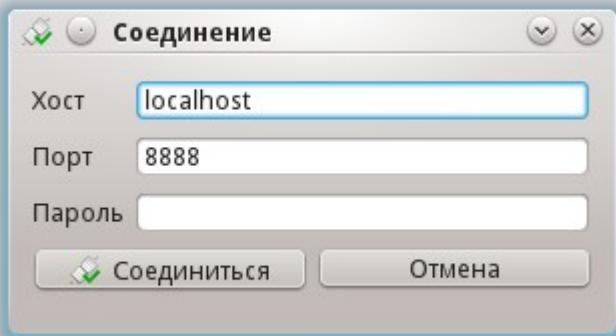


Рис. 5 Установка соединения с сервером утилит

В поле *Хост* введите адрес хоста сервера утилит, в поле *Порт* номер порта, который прослушивает сервер утилит, в поле *Пароль* введите пароль, если он используется для дополнительного шифрования закрытого ключа. Затем нажмите на кнопку *Соединиться*.

В случае успешного соединения, в главном окне появятся доступные на данном сервере утилит методы (см. раздел "Выполнение методов на сервере утилит").

Панель меню

После установления соединения с сервером утилит верхнее меню приобретает вид, показанный на предыдущем изображении и состоит из кнопок: "Назад" (кроме главного меню), "Процессы", "Сессия", "Отсоединиться", "Настройки" и "Помощь".

Возврат в Главное меню

Для возврата в главное меню служит кнопка "Назад" в главном меню. При отображении главного меню кнопка не отображается.

Запущенные процессы

Кнопка "Процессы" в панели меню покажет информацию о запущенных процессах. К работающему процессу можно подключиться, прервать его выполнение или просмотреть результат работы.

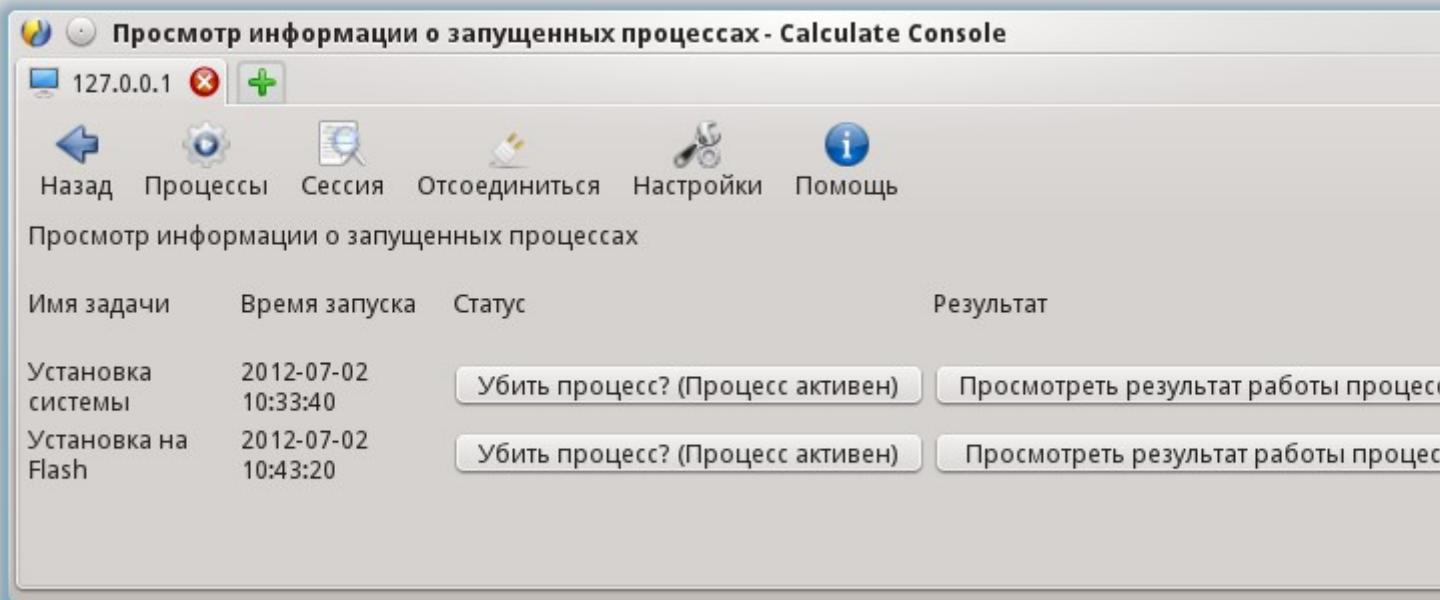


Рис. 6 Просмотр информации о запущенных процессах

Информация о сессии

Кнопка "Сессия" в панели меню отобразит окно с информацией о текущей сессии с возможностью очистить на сервере кэш процессов, принадлежащих данной сессии.

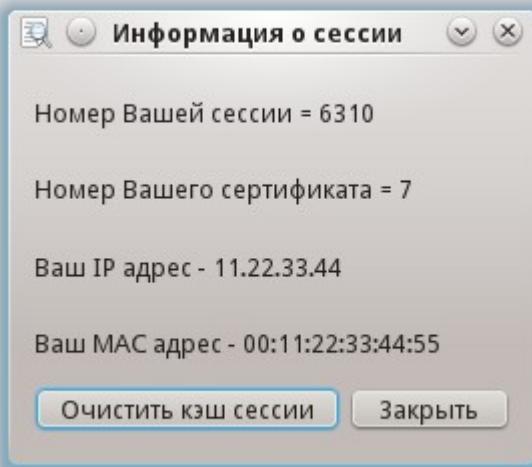


Рис. 7 Информация о текущей сессии Кнопка "Отсоединиться" в панели меню отобразит диалоговое окно закрытия сессии и тремя действиями:

- "Да" - отсоединиться и закрыть сессию.
- "Нет" - Отсоединиться, но не закрывать сессию.
- "Отмена" - не отсоединяться.

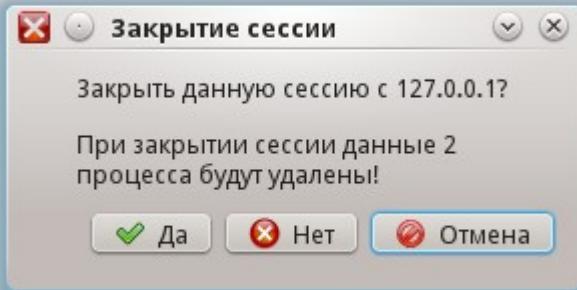


Рис. 8 Закрытие сессии

Настройка программы

Кнопка "Настройки" в панели меню служит для вызова окна настроек параметров Calculate Console и состоит из вкладок "Настройки интерфейса" и "Прочие настройки". Вкладка "Настройки интерфейса" включает поля:

- "*Расширенный просмотр*" - устанавливает режим просмотра результатов работы процесса: расширенный или сокращённый.
- "*Высота изображений*" - устанавливает фиксированную высоту изображений (0 - убрать изображения).
- "*Количество строк в таблице*" - устанавливает максимальное количество строк в таблицах, находящихся в результатах работы процесса и отображающих списки (например, список запросов на подпись сертификата).

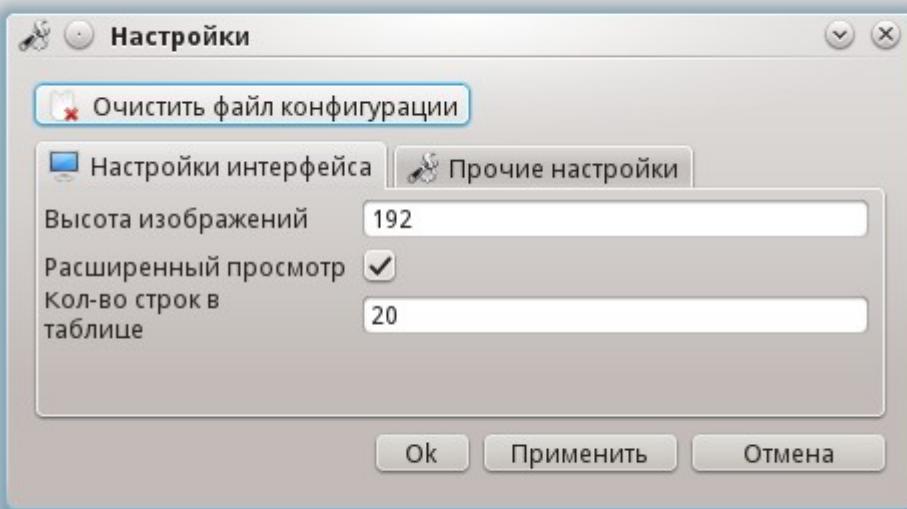


Рис. 9 Настройка интерфейса Вкладка "Прочие настройки" включает поля:

- "*Выбор языка*" - устанавливает язык интерфейса.
- "*Директория с сертификатами*" - устанавливает директорию, где находятся сертификаты клиента (при хранении их не в стандартной директории или на внешнем носителе).

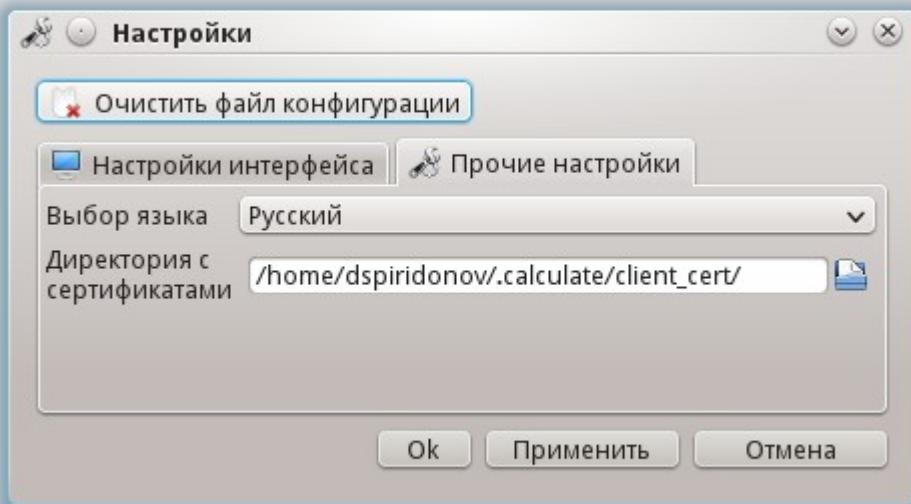


Рис. 10 Прочие настройки программы

Помощь

Кнопка "Помощь" в панели меню отобразит подменю из пунктов:

- "О программе" - выведет окно с информацией о Calculate Console
- "Справка" - откроет веб-страницу с этим руководством.
- "Сообщить об ошибке" - отобразит окно для отправки сообщения об ошибке разработчикам:

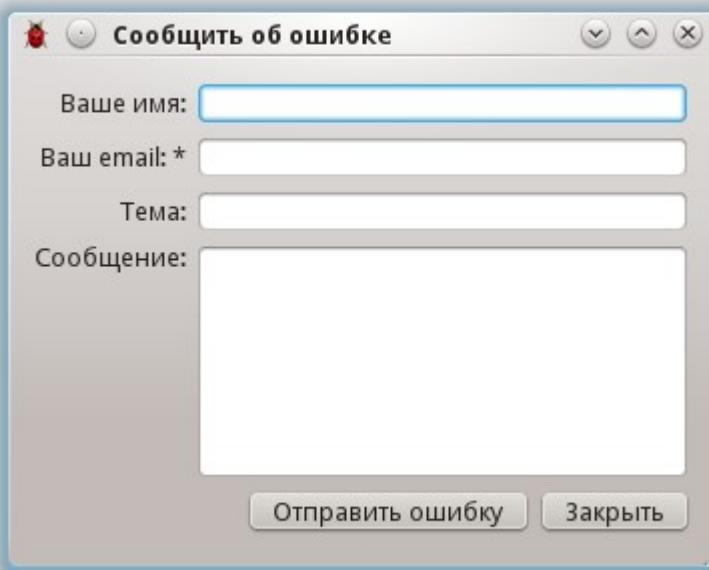


Рис. 11 Окно отправки сообщения об ошибке

Выполнение методов на сервере утилит

После соединения с сервером утилит в Главном окне появятся доступные для данного сертификата методы:

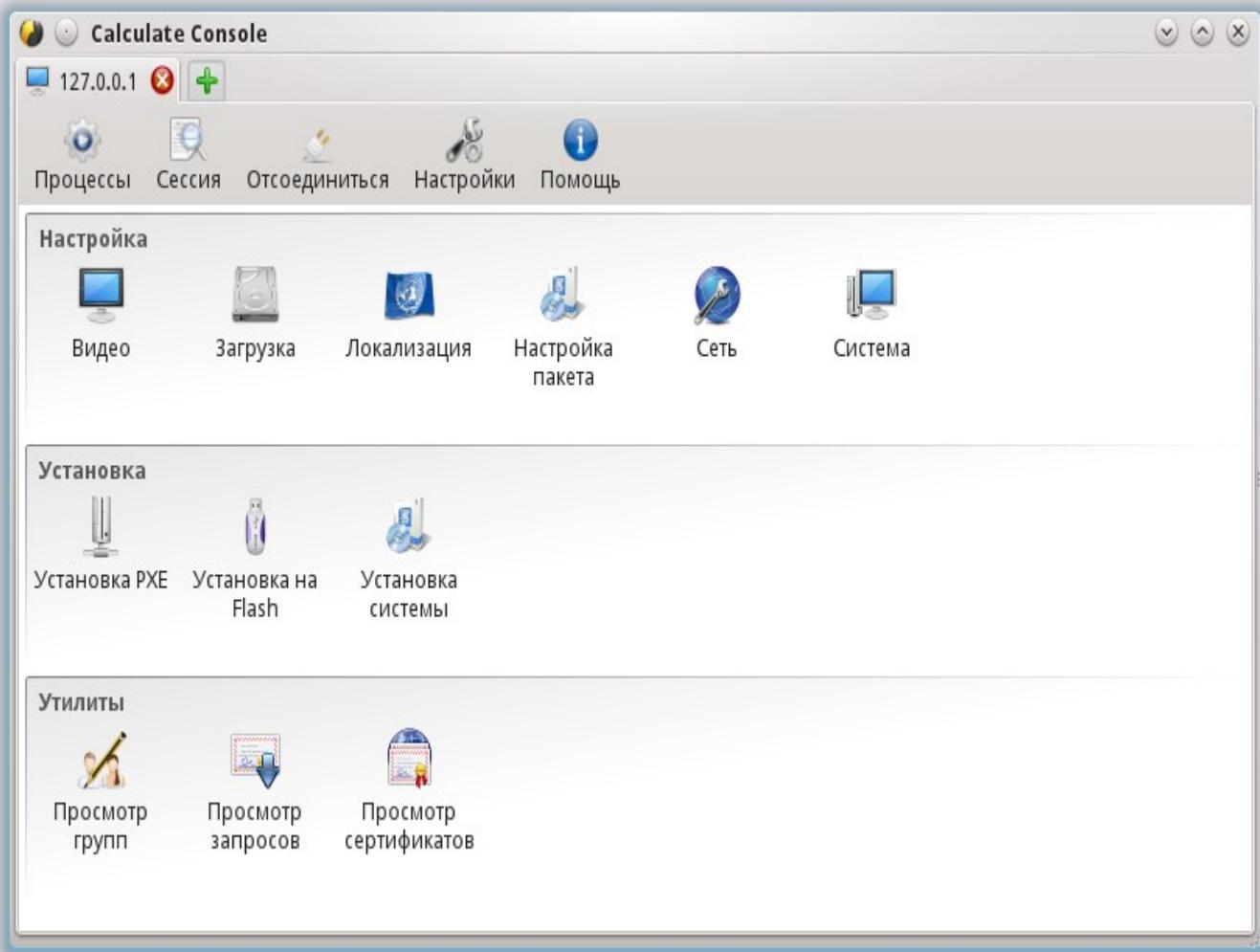


Рис. 12 Главное окно с доступными методами

Все методы разделены по категориям. При открытии метода откроется окно с необходимыми данными. Методы могут состоять из одного шага:

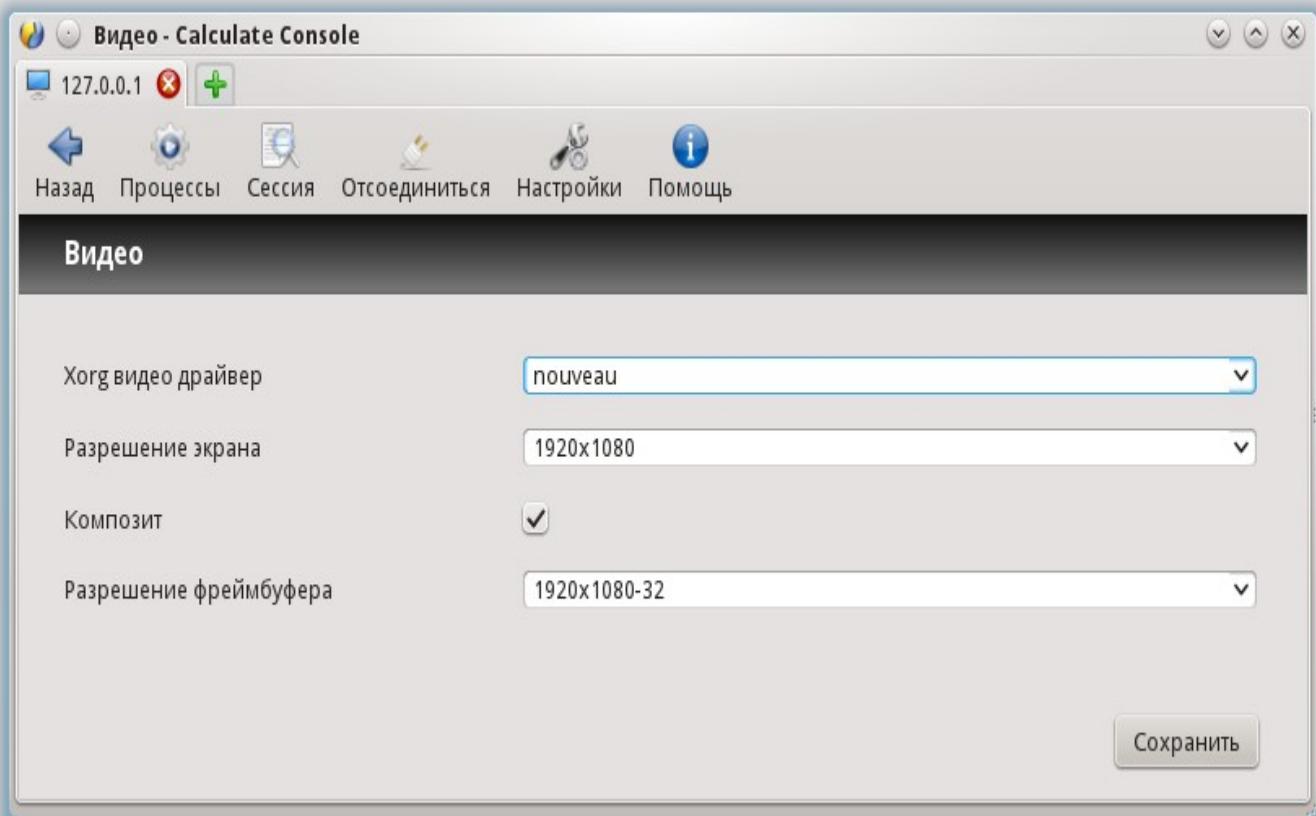


Рис. 13 Метод настройки видеокарты

Или из нескольких шагов:

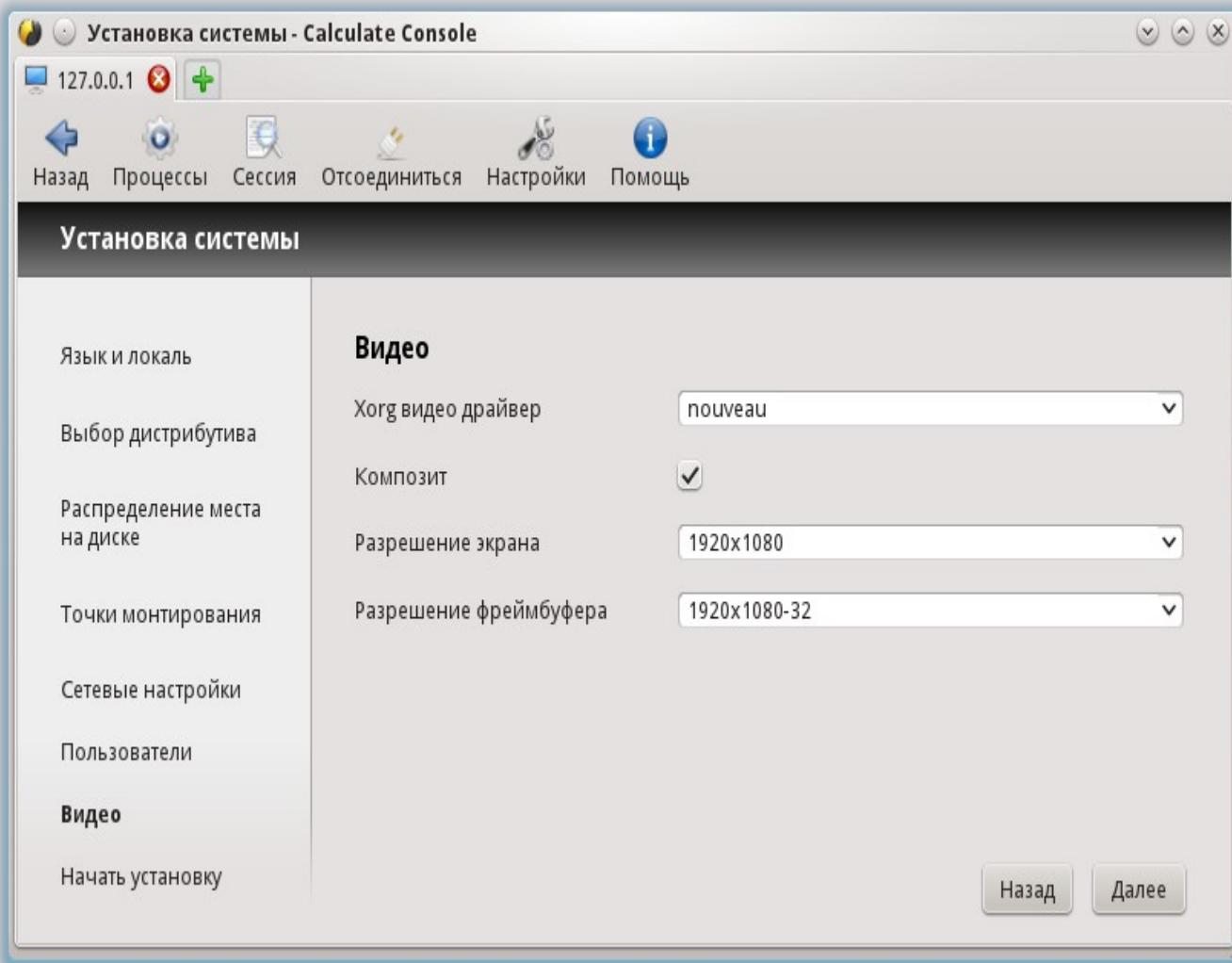


Рис. 14 Настройка видеокарты, как один из шагов метода установки системы

По шагам можно перемещаться, используя кнопки "Назад" и "Далее" или выбрав нужный шаг в списке слева. Начать выполнение метода можно только из последнего шага.

При выполнении метода откроется окно с результатами работы процесса. Во время выполнения, процесс можно прервать, нажав на кнопку "Прервать процесс".

По завершении работы процесса можно вернуться назад с помощью кнопки "Назад" в верхнем меню или закрыть процесс, нажав на кнопку "Закрыть". В этом случае данные о его работе будут удалены на сервере утилит и не будут отображаться в окне "Просмотр информации о запущенных процессах".

Безопасность и разграничение прав

Как уже упоминалось, для взаимодействия графического клиента с сервером утилит Calculate используется протокол "https", который позволяет зашифровывать передаваемые данные, используя сертификаты и ключи. Процесс получения сертификата описан ранее в разделе "*Создание сертификата*". В данном разделе речь пойдёт о подписании запросов и разграничении прав доступа по сертификатам.

Запросы

Запрос на подпись сертификата создаётся на стороне клиента после создания открытого и секретного ключей, и передаётся серверу утилит для подписания сертификата.

В графическом клиенте для работы с запросами на сервере утилит используется метод "**Просмотр запросов**" в категории утилиты.

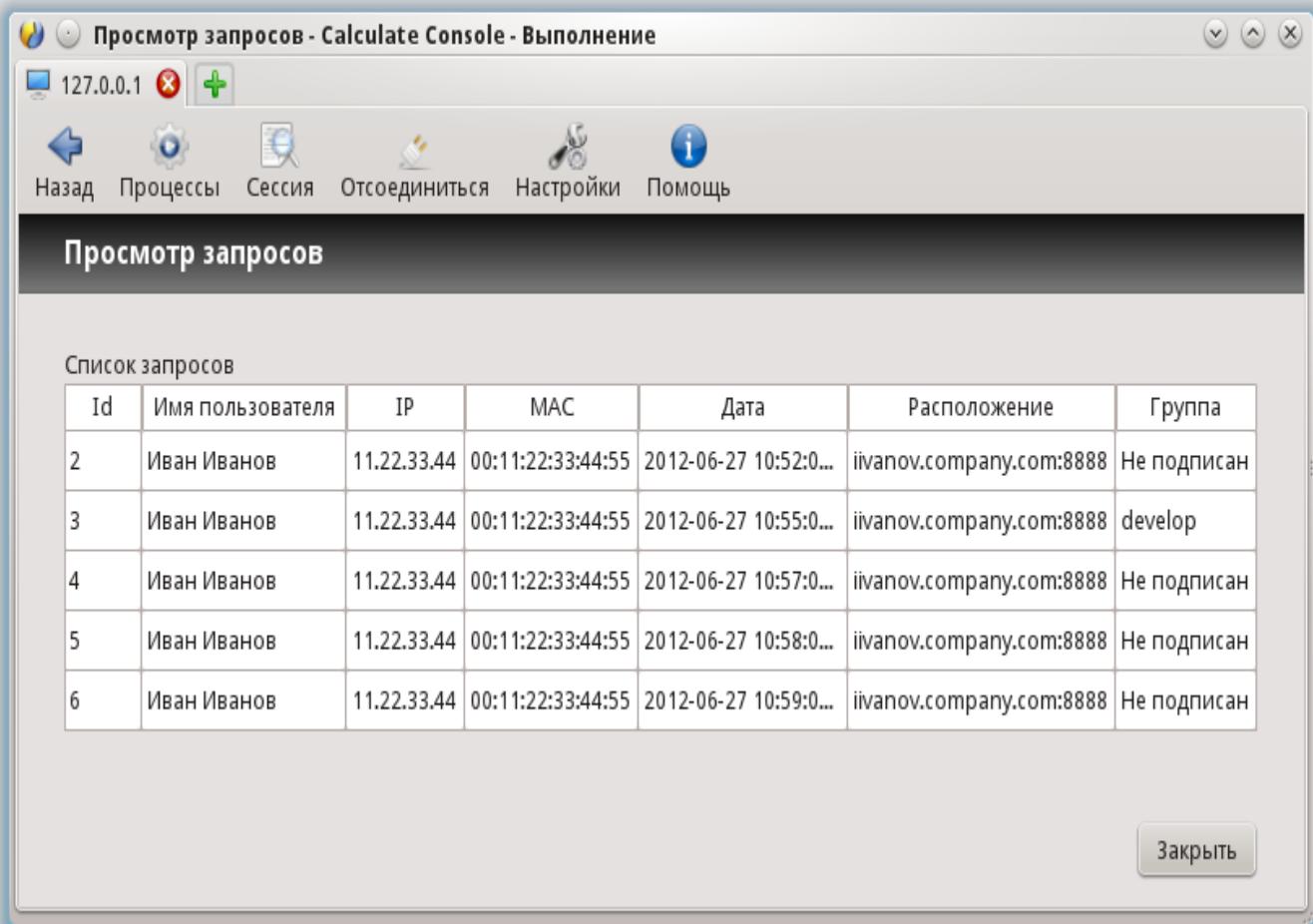


Рис. 15 Просмотр запросов

Для выбора запроса необходимо нажать на строку таблицы, после чего откроется окно с информацией о запросе, где можно изменить номер запроса (при этом данные о запросе не обновляются) и группу будущего сертификата.

Выбранный запрос можно подписать "Подтвердить" или отказать в подписании "Удалить".

Подписанный запрос, сертификат которого не забран клиентом, будет отображаться в таблице "Просмотр запросов" с называнием группы в столбце "Группа".

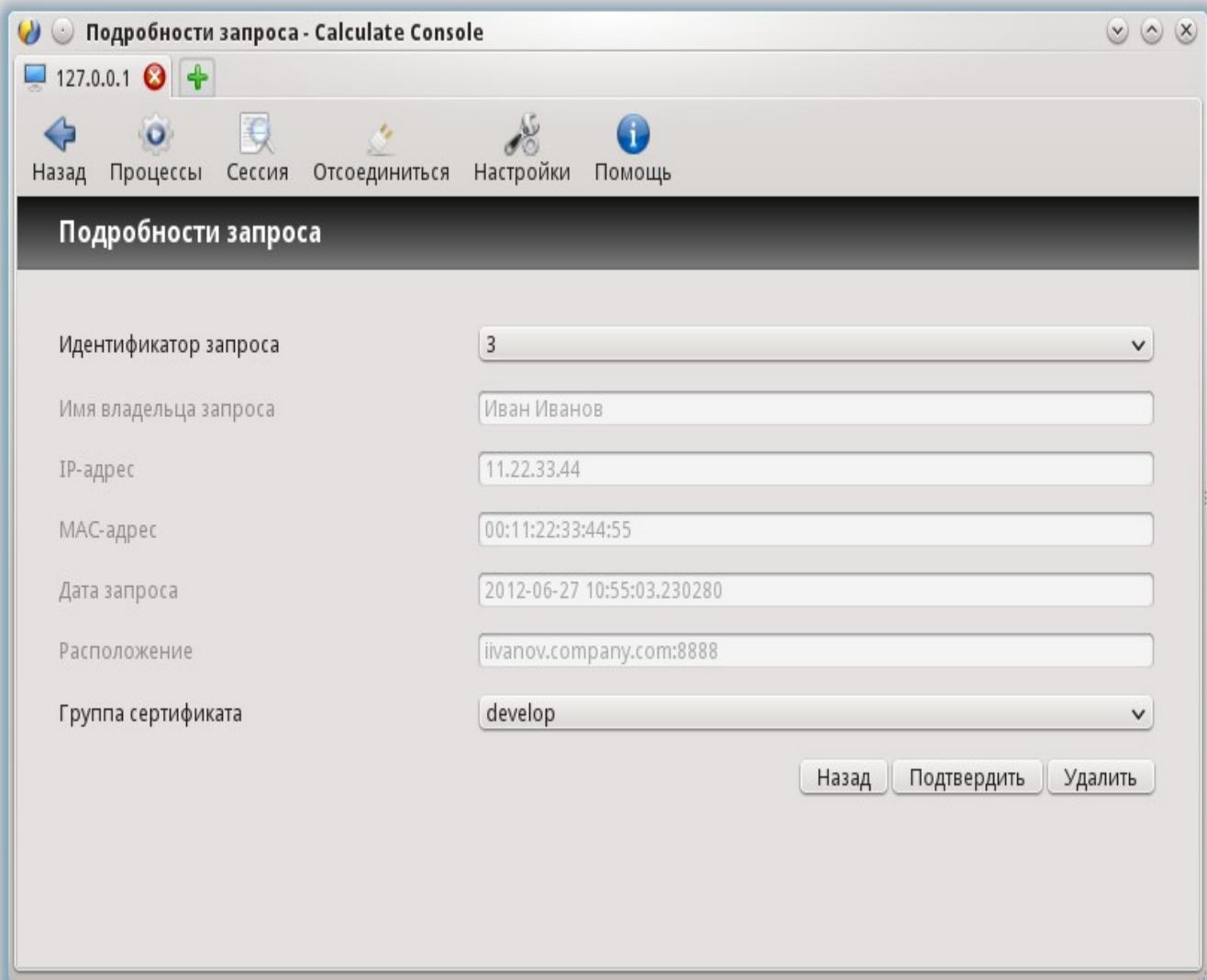


Рис. 16 Просмотр запросов

Сертификаты

Сертификат подтверждает соответствие между открытым ключом и информацией, идентифицирующей владельца ключа. Кроме того, в утилитах Calculate сертификаты используются для разграничения прав доступа с помощью указания в них группы прав.

При генерации сертификата для него устанавливается группа прав. Права записываются в одно из полей сертификата и не подлежат изменению.

Права групп

Права групп служат для разграничения прав на сервере утилит и указываются в сертификате клиента. Каждой группе соответствует список прав, к которым данная группа имеет доступ.

Права для групп сертификатов по умолчанию хранятся в файле `/var/calculate/server/conf/group_right.conf` в следующем виде: `group right1[, right2[, right3...]]`, например

```
manager install, get-sessions, request
user get-sessions, request, view_cert
```

Для изменения прав конкретного сертификата используется файл `/var/calculate/server/conf/right.conf`, куда необходимо вписать права и номера сертификатов.

Пример:

```
install 1 2 -3
```

Для сертификатов с номерами 1 и 2 добавить право на метод "install" и удалить его для сертификата с номером 3.

Права для конкретного сертификата имеют приоритет перед правами для группы сертификата.

Для управления правами групп на сервере утилит в клиенте используется метод "**Просмотр групп**" в категории утилиты.

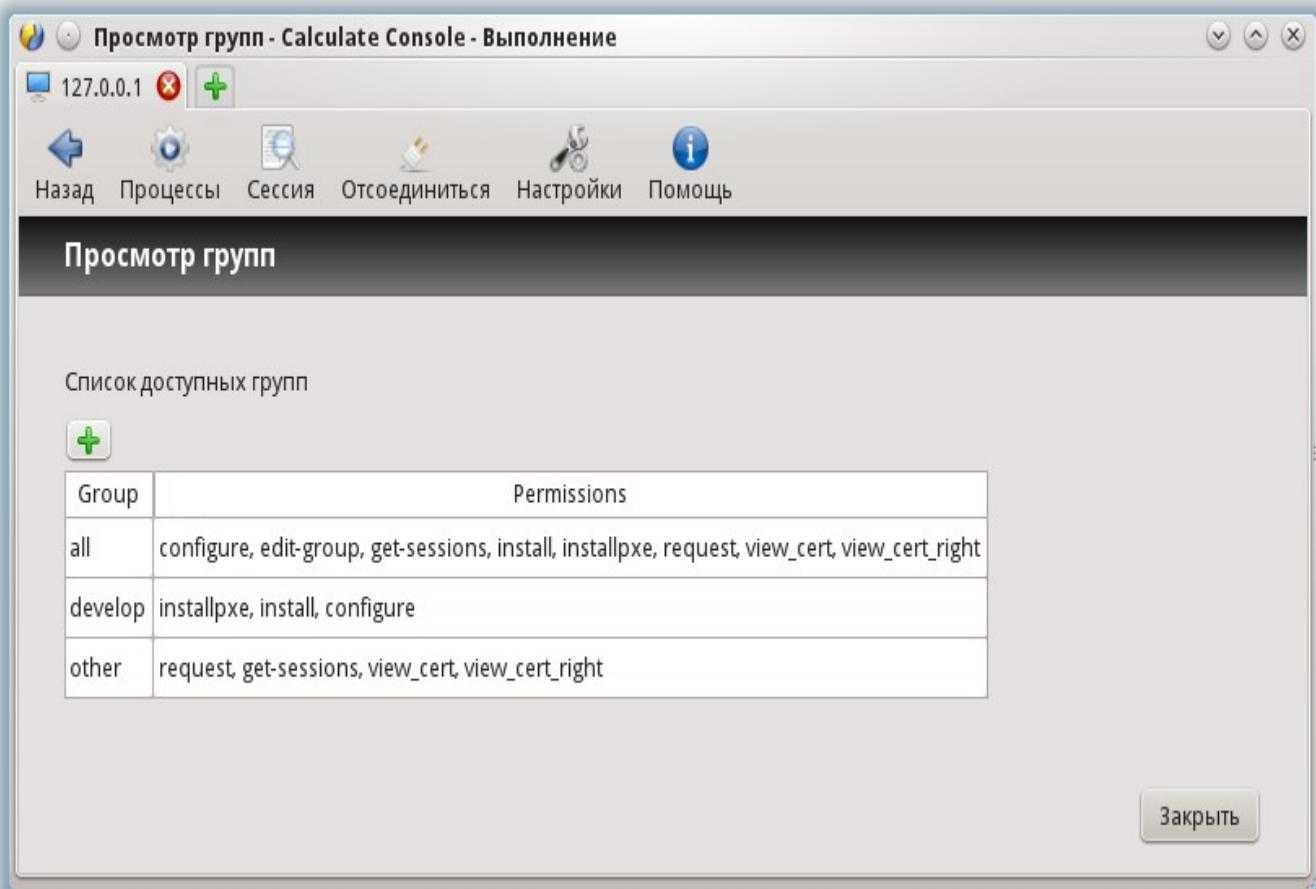


Рис. 17 Просмотр групп

Для добавления новой группы нажмите на кнопку "+" (плюс). В открытом действии "Добавление группы" введите имя создаваемой группы в поле "Имя группы" и выберите права для группы в поле "Права группы".

Для выбора группы необходимо нажать на строку таблицы, после чего откроется "Подробности группы", где можно изменить права либо удалить группу.

Консольный клиент утилит Calculate

- [Консольный клиент утилит Calculate](#)
- [Введение](#)
- [Создание сертификата](#)
- [Выполнение методов на сервере утилит](#)

- [Действия с клиентом](#)
- [Работа с сессией](#)
- [Работа с процессами](#)
- [@cl-consoled@](#)

Введение

Клиент утилит Calculate Console служит для сетевого доступа по протоколу <https> к функциям сервера утилит Calculate 3 ([calculate-core](#)). Установить клиент можно при помощи пакета `SYStem-apps/calculate-console`.

Создание сертификата

Для работы с сервером утилит необходимы секретный ключ и сертификат, подписанный сервером утилит.

Для генерации секретного ключа и создания запроса на подпись сертификата, используйте ключ `--gen-cert-by <хост>` и ключ `--port <порт>` для указания номера порта (по умолчанию порт 8888), например:

```
cl-console --gen-cert-by 192.168.0.56 --port 8888
```

Если по адресу 192.168.0.56 порт 8888 прослушивает сервер утилит, то пользователю будет предложено создать запрос на подпись сертификата.

Первым действием является установка пароля для дополнительного шифрования секретного ключа. Введённый пароль будет запрашиваться для установки соединения с сервером утилит. Чтобы не зашифровывать секретный ключ, оставьте поле пароль пустым.

Также необходимо ввести некоторые данные, необходимые для создания сертификата. В квадратных скобках будут показаны значения по умолчанию, например:

Имя хоста [`iivanov@iivanov.company.ru`] :

Имя пользователя [Иван Иванов] :

Большая часть данных, кроме названия организации и города, будет заполнена автоматически. Не меняйте автоматически подставленные значения, если не уверены, что значения неверны.

После отправления запроса на сервер утилит, пользователю будет сообщён номер, под которым запрос находится на текущем сервере утилит.

Подписать запрос можно на сервере утилит через клиента или с помощью команды:

```
cl-core --sign-client <n>
```

где - номер запроса.

После подписания сертификата на сервере, его необходимо забрать, используя ключ `--get-cert-from <хост>` и ключ `--port <порт>`, например:

```
cl-console --get-cert-from 192.168.0.56 --port 8888
```

Если запрос ещё не подписан или отвергнут, то появится соответствующее сообщение.

Для указания пути к файлам сертификата и секретного ключа (при нахождении их не в стандартной директории) используйте ключ `--cert-path <путь>`:

```
cl-console --cert-path /media/flash/cert_dir
```

Выполнение методов на сервере утилит

Для просмотра доступных методов на сервере утилит используйте команду

```
cl-console --host <хост> --port <порт>
```

По умолчанию клиент соединяется с локальным сервером утилит и 8888 портом, например:

```
cl-console  
cl-console --host 192.168.0.56 --port 9999
```

Первая команда выведет все доступные методы на локальном сервере утилит, прослушивающем порт 8888, вторая - на сервере утилит, находящемуся по адресу 192.168.0.56:9999.

Для запуска метода используйте ключ `--method <метод>`, например:

```
cl-console --method install --iso /path_to_image/cld-x86_64.iso -d  
/dev/sda1:swap -d /dev/sda2:/ext4:on
```

Для просмотра справки метода используйте ключ `-h, --help`:

```
cl-core --method install --help
```

Ключ `-f, --force` устанавливает режим, при котором пользователю не задаются вопросы и не отображаются предварительные параметры.

Ключ `--no-progress` отключает отображение прогрессбаров (текущего прогресса выполнения задачи).

Действия с клиентом

Работа с сессией

Для просмотра информации о сессии и сертификате используйте ключ `--session-info`:

```
cl-console --session-info
```

Для просмотра списка активных сессий на сервере утилит используйте ключ `--session-list`:

```
cl-console --session-list
```

Для просмотра информации о конкретной сессии, используйте ключ `--session-num-info`, например, команда

```
cl-core --session-num-info 5474
```

выведет информацию о сессии с номером 5474.

Для очистки кэша своей сессии используйте ключ `--session-clean`:

```
cl-console --session-clean
```

Работа с процессами

Каждый запущенный метод на сервере утилит является процессом, результат работы которого можно просмотреть.

Для просмотра списка запущенных процессов текущей сессии, используйте ключ `--list-pid`:

```
cl-core --list-pid
```

Для более подробного вывода совместно с ключом `--list-pid` используйте ключ `-d, --dump`:

```
cl-core --list-pid --dump
```

Для просмотра результатов работы процесса, используйте ключ `--pid-result <pid>`, например

```
cl-core --pid-result 1234
```

выведет результат работы процесса с номером 1234.

Если необходимо прервать работающий процесс, используйте ключ `--pid-kill <pid>`, например
`cl-core --pid-kill 1234`

сообщит серверу утилит, что необходимо завершить процесс с номером 1234.

cl-consoled

Если пользователем используется пароль для дополнительного шифрования секретного ключа, то консольным клиентом используется демон `cl-consoled`.

`cl-consoled` предназначен для кратковременного хранения введённых паролей, чтобы пользователю не было необходимо при каждом соединении вводить пароль. Пароль к секретному ключу сервера утилит выдаётся только при совпадении ряда данных.

Для остановки `cl-consoled` используйте команду

```
cl-console --stop-consoled
```

В этом случае при следующем запуске `cl-console` заново потребует пароль и запустит `cl-consoled`.

При необходимости обновить список отзыва сертификатов, используйте ключ `--update-crl`:

```
cl-console --update-crl
```

Сервер утилит Calculate

- [Сервер утилит Calculate](#)
- [Введение](#)
- [Начало работы, создание сертификата](#)
- [Подписание сертификата у другого сервера утилит](#)
- [Создание самоподписного сертификата](#)
- [Запуск сервера утилит](#)
- [Действия с сервером утилит](#)
- [Операции с запросами и сертификатами](#)
- [Локальный запуск процессов](#)
- [Другие действия](#)
- [Стандартные методы сервера утилит](#)

Введение

Сервер утилит Calculate Core служит для выполнения методов утилит, таких как установка, настройка системы и т.д. и для осуществления сетевого доступа клиентам (`cl-console` и `cl-console-gui`) по протоколу <https> к функциям утилит Calculate. Calculate Core входит в состав утилит Calculate 3. Установить сервер утилит можно при помощи пакета `sys-apps/calculate-core`.

Начало работы, создание сертификата

Для запуска сервера утилит необходимо создать сертификат одним из двух способов:

- создать запрос на подписание сертификата (далее запрос) и подписать сертификат у другого сервера;
- создать самоподписной корневой сертификат и использовать его в качестве сертификата сервера.

Для запуска сервера утилит необходим серверный сертификат. Корневой самоподписной сертификат необходим для подписания сертификатов другим серверам утилит.

Подписание сертификата у другого сервера утилит

Для генерации секретного ключа и запроса, а так же для отправки запроса на сервер утилит используйте команду

```
cl-core --gen-cert-by HOST --port PORT
```

, где HOST - сетевой адрес сервера утилит, PORT - порт, который прослушивает сервер утилит (по умолчанию 8888)

Например,

```
cl-core --gen-cert-by 192.168.0.123 --port 4567
```

После подписания сертификата на сервере утилит необходимо забрать его с помощью команды

```
cl-core --get-cert-from ROOT_HOST --port PORT
```

, где ROOT_HOST - сетевой адрес сервера утилит, PORT - порт, который прослушивает сервер утилит (по умолчанию 8888)

Например,

```
cl-core --get-cert-from 192.168.0.123 --port 4567
```

Создание самоподписанного сертификата

Для создания самоподписанного корневого сертификата используйте команду

```
cl-core --gen-root-cert
```

Для использования созданного самоподписанного корневого сертификата как сертификат сервера утилит используйте команду

```
cl-core --use-root-as-server
```

Запуск сервера утилит

После генерации сертификата сервер утилит можно запустить с помощью команды

```
cl-core --start
```

При этом сервер утилит будет прослушивать порт 8888, для указания порта используйте ключ --port, например

```
cl-core --start --port 5648
```

Для запуска в режиме отладки (debug), используйте ключ --debug

Действия с сервером утилит

Операции с запросами и сертификатами

Просмотр запросов и сертификатов

Многие действия с сервером утилит можно выполнять непосредственно на сервере, без использования клиентов. Для этого необходимы права суперпользователя - root.

Для просмотра клиентских запросов используйте ключ --show-request с указанием номера запроса или слова "all" для просмотра списка запросов, например

```
cl-core --show-request all
```

```
cl-core --show-request 2
```

Для просмотра клиентских подписанных сертификатов используйте ключ `--cert` с указанием номера сертификата или слова "all" для просмотра списка сертификатов, например

```
cl-core --cert all  
cl-core --cert 2
```

С помощью ключа `--dump` можно просмотреть сразу все сертификаты

```
cl-core --cert all --dump
```

или содержимое файла сертификата, например

```
cl-core --cert 4 --dump
```

Для просмотра запросов и сертификатов серверов утилит используйте ключ `--server-cert` с указанием номера или "all", например

```
cl-core --server-cert all  
cl-core --server-cert 2
```

Подписание запросов и отзыв сертификатов клиента

Для подписание клиентского запроса на подпись сертификата используйте ключ `--sign-client` с указанием номера запроса, например

```
cl-core --sign-client 4
```

После этого введите группу прав для нового сертификата (изменить её в дальнейшем невозможно).
Подписание осуществляется серверным сертификатом.

Если необходимо отвергнуть клиентский запрос, используйте ключ `--del-client-req` с указанием номера запроса, например

```
cl-core --del-client-req 4
```

Для удаления уже подписанного сертификата клиента совместно с ключом `--cert` и указанием номера сертификата используйте ключ `--remove`, например

```
cl-core --cert 4 --remove
```

Также создать сертификат пользователь с правами группы "all" можно с помощью команды

```
cl-core --bootstrap username
```

например, с помощью команды

```
cl-core --bootstrap iivanov
```

для пользователя `iivanov` будет создан сертификат с правами группы "all" и добавлен в доверенные сертификат сервера утилит.

Для удаления всех сертификатов, запросов и конфигурационных файлов на сервере совместно с ключом `--bootstrap` используйте ключ `--remove-certs`, например:

```
cl-core --bootstrap iivanov --remove-certs
```

Подписание запросов и отзыв сертификатов сервера утилит

Для подписание запроса на подпись сертификата от другого сервера утилит используйте ключ `--sign-server` с указанием номера запроса, например

```
cl-core --sign-server 4
```

Подписание осуществляется корневым сертификатом.

Если необходимо отвергнуть запрос сервера утилит, используйте ключ --del-server-req с указанием номера запроса, например

```
cl-core --del-server-req 4
```

Для отзыва подписанного сертификата сервера утилит (добавление в список отзыва) используйте ключ --revoke-cert с указанием номера сертификата, например

```
cl-core --revoke-cert 4
```

Для удаления списка отзывов сертификатов, используйте команду

```
cl-core --revoke-cert rm
```

Изменение прав сертификатов

Права для групп сертификатов по умолчанию хранятся в файле

/var/calculate/server/conf/group_right.conf в следующем виде: group right1[, right2[,right3...]], например

```
manager install, get-sessions, request  
user get-sessions,request,view_cert
```

Для изменения прав конкретного сертификата используется файл

/var/calculate/server/conf/right.conf, куда необходимо вписать права и номера сертификатов, например

```
install 1 2 -3
```

для сертификатов с номерами 1 и 2 добавить право на действие install и удалить его для сертификата с номером 3.

Права для конкретного сертификата имеют приоритет перед правами для группы сертификата.

Для изменения прав конкретного сертификата клиента используется ключи --right-add и --right-del совместно с ключом --cert, например

```
cl-core -c 6 --right-del install_pxe,install  
cl-core -c 7 --right-add install_pxe,install,configure_video
```

В примере для сертификата с номером 6 установится запрет на методы, требующие права install_pxe и install, а для сертификата с номером 7 установится разрешение на методы install_pxe, install и configure_video.

Локальный запуск процессов

Запуск методов с помощью ключа --method

Все действия на сервере утилит можно запускать как через клиентов ([cl-console-gui](#), [cl-console](#)), с использованием шифрования по сертификатам и позволяющим выполнять действия на удалённых серверах утилит, так и с помощью самих серверов утилит, выполняющих действия напрямую и только на локальных серверах утилит.

Для просмотра всех доступных действий на сервере утилит используйте команду

```
# cl-core --list-methods
```

```
install - Установка системы  
setup_boot - Загрузка  
core_setup - Настройка пакета
```

...

Для запуска метода используйте команду

```
cl-core --method METHOD
```

например:

```
cl-core --method install --iso /path_to_image/cld-x86_64.iso -d  
/dev/sda1:swap -d /dev/sda2:/ext4:on
```

Для просмотра справки действия используйте команду

```
cl-core --method METHOD --help
```

например:

```
cl-core --method install --help
```

Ключ -f, --force устанавливает режим, при котором пользователю не задаются вопросы и не отображаются предварительные параметры (brief).

Ключ --no-progress отключает отображение прогрессбаров (текущего прогресса выполнения задачи).

Ссылки на методы

При установке системы для всех действий на сервере утилит создаются символические ссылки вида cl-method, где method - название метода, которое можно получить с помощью cl-core --list-methods.

Например, для метода setup_network ссылкой будет команда cl-setup-network (символы "_" заменяются на "-").

Для создания отсутствующих ссылок и удаления ссылок на отсутствующие действия сервера утилит используйте команду

```
cl-core --create-symlink
```

Выполнять команду cl-core и все символические ссылки на неё может только пользователь root.

Все символические ссылки работают по принципу вызова метода сервера утилит - только на локальной машине без использования сертификатов и шифрования.

Другие действия

Проверить конфигурацию (наличие сертификата и секретного ключа, их соответствие друг другу, а также действителен ли сертификат) можно с помощью ключа --check

Указать путь к для ведения журнала событий (логов) можно с помощью ключа --log-path, например:

```
cl-core --log-path /var/log/calculate/mylogs/
```

Файл логов по умолчанию - /var/log/calculate/logging_cl_core.out.

Указание пути к PID файлу осуществляется с помощью ключа --pid-file PIDFILE Дополнительно есть две опции для отображения прогресса выполнения в GUI:

- --gui-progress - отображает индикатор прогресса в GUI
- --gui-warning - отображает предупреждения в конце

Передача пароля со стандартного ввода осуществляется с помощью ключа -P, например:

```
cat pass | cl-core -P --method install -u test -f
```

Опция -P должна использоваться вместе с опцией -f, т.к. при перенаправлении потоков ввода/вывода использование интерактивного режима невозможно.

Стандартные методы сервера утилит

Все методы сервера утилит разделены на категории. В версии сервера утилит 3.2.0 существуют следующие стандартные категории: Клиент, Настройка, Обновить, Рабочий стол, Установка и Утилиты.

Клиент

Категория Клиент включает в себя методы для изменения поведения системы(локальная/доменная) и для смены паролей пользователей.

Настройка

Категория Настройка включает в себя методы для настройки параметров системы и пакетов.

Обновить

Категория Обновить включает в себя методы для обновления системы, настроек и смены профиля.

Рабочий стол

Категория Рабочий стол включает в себя методы для принудительного выхода пользователей из сеанса и настройки профилей пользователей.

Установка

Категория Установка включает в себя методы для установки системы.

Утилиты

Категория Утилиты включает в себя методы для работы с сертификатами, запросами и группами прав сертификатов.

Шаблоны Calculate

- [Шаблоны Calculate](#)
- [Введение](#)
- [Особенности работы с шаблонами](#)
- [Шаблоны установки](#)
- [Форматы файлов](#)
- [Управляющие элементы](#)
- [Способы объединения](#)
- [Расположение](#)
- [Правила именования файлов](#)
- [Правила именования директорий](#)
- [Права доступа](#)
- [Символические ссылки](#)
- [Схема объединения](#)
- [Правила объединения](#)
- [Правила объединения, действующие по умолчанию](#)
- [Изменение правил объединения](#)
- [Формат CXmlConf](#)
- [Области](#)
- [Переменные](#)
- [Списки](#)
- [Разделённые списки](#)
- [Комментарии](#)
- [Управляющие элементы](#)
- [Формат XML](#)

- [Поддерживаемые форматы XML](#)
- [Отличия от формата CXmlConf](#)
- [Объединение элементов XML](#)
- [Примеры использования](#)
- [Формат 'patch'](#)
- [Особенности.](#)
- [Описание.](#)
- [Формат 'diff'](#)
- [Формат 'kernel'](#)
- [Формат 'dconf'](#)
- [Формат 'json'](#)

Введение

Традиционно в ОС Linux настройки приложений хранятся в текстовых файлах, как правило в директории `/etc`, реже в `/var`. Форматы таких конфигурационных файлов различаются: от простых "переменная=значение", до более сложных С-подобных конструкций, либо составленных в XML.

Простой на первый взгляд подход обеспечивает исключительную стабильность в работе приложений, так как за сохранность настроек отвечает исключительно файловая система.

Различные дистрибутивы Linux, как правило, предлагают свои программы настройки приложений. К сожалению, такой подход, имея неоспоримое преимущество в удобстве, имеет также ряд недостатков:

- пользователь привязывается к определенной программе настройки (дистрибутиву);
- количество настроек, как правило, ограничено интерфейсом программы;
- прямое редактирование настроек становится затруднительным, т.к. программа переписывает файл при внесении изменений.

Особенности работы с шаблонами

В утилитах Calculate 2 и 3 подход к шаблонам был существенно переработан и обладает рядом отличий от первой версии:

- Основной метод переноса шаблонов - объединение с конфигурационными файлами системы. При этом поддерживаются все популярные форматы файлов.
- Объединение шаблона с конфигурационным файлом производится посредством конвертации в XML формат. При этом формат файла шаблона может отличаться от конфигурационного файла.
- Файл шаблона может содержать заголовок, описывающий методы объединения.
- Имена встроенных переменных переименованы согласно типу.

Шаблоны установки

Программа Calculate заменяет прямое редактирование файлов настроек на создание шаблонов.

Форматы файлов

Согласно методу хранения данных, файлы шаблонов могут иметь один из перечисленных форматов:

- **apache, kde, bind, postfix, proftpd, samba, procmail, ldap, dovecot, xml_xfce, xml_xfcepanel, xml_gconf, xml_gconf_tree, compiz, plasma, squid, dhcp, openrc** - форматы файлов настроек распространенных приложений
- **bin** - двоичный формат файлов
- **raw** - сырой текст
- **patch** - шаблон для применения регулярных выражений (использует специальный вид объединения *patch*).

Для формата **kde** не обрабатываются параметры '-' '+' для элементов внутри области.

Названия параметров в **openrc**-формате нечувствительны к регистру букв.

Управляющие элементы

Помимо настроек сервисов, записанных в оригинальном формате программы, файлы шаблонов содержат служебные записи, которые условно можно разделить на несколько типов.

Переменная

Переменная - текстовый элемент в шаблоне, имеющий имя, который заменяется в соответствующем конфигурационном файле значением. Переменная имеет имя, значение, область действия.

- имя - латинские буквы и цифры
- значение - текст для замены (создается в программе)
- область действия - действует глобально для всех шаблонов или локально для одного

Переменные подразделяются на переменные шаблонов и переменные функций.

У переменных шаблонов глобальная область действия, то есть любая переменная шаблона, доступная в программе, может быть использована в любом шаблоне. Значение переменной нельзя изменить в шаблоне.

Переменные функций могут быть созданы в шаблоне; также можно изменить значение переменной в шаблоне. Область действия переменной функции - текущий шаблон.

Для передачи значений переменных функций из текущего шаблона в другой шаблон используется стек переменных функций шаблонов (LIFO), в который при помощи функции шаблонов `push()` записывается значение из текущего шаблона, а функцией шаблона `pop()` получаем значение в другом шаблоне.

Стек переменных функций шаблонов

Стек (LIFO - "последним пришел, первым вышел") для хранения значений переменных функций. Доступен глобально для всех шаблонов, для работы используются функции шаблонов, `push()` - запись, `pop()` - чтение. Запись и чтение возможны как в одном шаблоне, так и в разных.

Заголовок

Заголовок - управляющая запись шаблона, определяющая методы переноса шаблона в систему. Заголовок шаблона записывается первой строкой файла и имеет следующий вид:

```
# Calculate параметр1=значение1 параметр2 [параметр3=значение3 . . . ]
```

Содержимое заголовка может быть разбито на строки. В этом случае в конце каждой строки заголовка, кроме последней, должен стоять знак "\\" (обратная косая черта).

Если заголовок отсутствует, настройки файла шаблона определяются исходя из принятых значений по умолчанию. **Допустимые параметры:**

Формат

- **format=[...]** - формат файла шаблона (см. форматы файлов). По умолчанию формат файла шаблона определяется как "raw", или "bin" для файлов, содержащих двоичные данные.
- **comment=[.]** - обозначение начала строкового комментария (пример: "#"). **Объединение**
- **append=[join|before|after|replace|remove|skip|patch|clear]** - способ объединения. По умолчанию способ объединения устанавливается в соответствии с форматом файла. Если append=skip - шаблон пропускается. Если append=clear, в случае шаблона файла конфигурационный файл будет очищен - длина файла 0, а в случае шаблона директории все файлы и директории внутри конфигурационной директории будут удалены.
- **force** - удалять существующие файлы перед записью конфигурационного файла. Правило действует по умолчанию, если указан параметр "symbolic".
- **link=путь** - путь к конфигурационному файлу, с которым объединяется файл шаблона. По умолчанию путь совпадает.
Пример: "link=/etc/conf.d/net.example"

- **path=путь** - путь к директории, в которой будет находиться конфигурационный файл
- **name=имя** - имя конфигурационного файла
- **mirror** - выполнять объединение только в случае существования конфигурационного файла. Если конфигурационный файл задан параметром "link" и он не существует, файл назначения удаляется.
- **protected (добавлено в 3.1.1)** - защитить файл при удалении пакета, которому он принадлежит.
- **symbolic** - создать символьическую ссылку на файл, указанный параметром "link".
- **autoupdate** - при установке любого пакета запускается специальный скрипт `cl-update-config`, который применяет шаблоны для устанавливаемого пакета. Конфигурационные файлы устанавливаемого пакета защищены. Иными словами, при установке пакета, если такой же файл существует в системе и изменен, он не будет заменен. Замена конфигурационных файлов установленных пакетов производится командой `dispatch-conf`. Если в заголовке шаблона есть параметр **autoupdate** то после применения шаблона конфигурационный файл будет скопирован в систему. Если параметр отсутствует то для переноса измененного конфигурационного файла необходимо использовать `dispatch-conf`.
- **run (добавлено в 3.1.1)** - оболочка для выполнения конфигурационного файла, полученного из шаблона. Сценарий выполняется сразу же после обработки этого шаблона.
Пример: "run=/bin/bash"
- **exec** - оболочка для выполнения конфигурационного файла, полученного из шаблона. Сценарий добавляется в очередь, которая выполняется после обработки всех неисполнимых шаблонов.
Пример: "exec=/bin/bash" (начиная с версии 3.1.1 выполняемый сценарий не сохраняется на диск)
- **merge** - выполнить в конце настройку перечисленных пакетов.
Пример: "merge=openrc,grub" (начиная с версии 3.1.1 пакеты необходимо указывать вместе с категорией `merge=sys-apps/openrc, sys-boot/grub`)
- **postmerge (добавлено в 3.1.10)** - выполнить в конце настройку перечисленных пакетов. В отличие от merge пакеты будут настроены после `pkg_postinst()` во время сборки пакета.
Пример: "postmerge=sys-apps/openrc,sys-boot/grub"
- **env - (добавлено в 3.1)** использовать в шаблоне переменные указанного модуля. Если модуль в системе отсутствует - шаблон будет пропущен.
Пример: "env=install" Права доступа
- **chmod=XXX** - права доступа к конечному файлу (пример: "644"). По умолчанию права соответствуют оригинальному файлу. Если его нет, права соответствуют файлу шаблона.
- **chown=user:group** - владелец и/или группа конечного файла (пример: "root:root"). Условия
- **переменная[>]**

Метки

Вы можете настроить систему исходя из *аппаратных требований* компьютера, *сетевых установок* и прочих условий. Для этого в файле шаблона вы можете вместо постоянных значений устанавливать *метки* переменных.

Пример файла `/etc/conf.d/hostname`:

```
HOSTNAME="#-os_net_hostname-#"
```

Для использования переменных из другого модуля перед переменной, через точку указывается модуль **(добавлено в 3.1)**:

Пример:

```
#-install.os_install_root_dev-#
```

Условные блоки

Файл шаблона может содержать *условные блоки*.

Условные блоки - выделенный текст шаблона, подставляемый в случае соответствия *регулярного выражения*.

Регулярное выражение - метод проверки значений выражений. В качестве значений могут применяться следующие типы данных:

- **переменные** - встроенные переменные Calculate.
- **числа** - целые и дробные числа, в качестве разделителя дробной части выступает точка ".".
- **строки** - буквы, цифры, специальные символы.
- **номера версий** - числа и одна или две точки, разделяющих номер версии (например, 8.5.1).

Методом проверки выступает арифметическая операция **>**,

Арифметические операции могут объединяться условием "И" (&) и "ИЛИ" (||). Приоритет в данном случае будет отдаваться условию "И".

Условный блок должен начинаться с метки "#?переменная1==значение1(...)#" и заканчиваться "#переменная1#", записанными в начале строки.

Пример условного блока файла /etc/make.conf:

```
#?os_arch_march==i686&&os_linux_shortname==CLD#
    CFLAGS="-O2 -march=i686 -pipe"
    CHOST="i686-pc-linux-gnu"
#os_arch_march#
```

В приведенном примере сравниваются переменные `setup_march` и `setup_sys_shortname` со строковыми значениями "i686" и "CLD" соответственно. В случае, если значения обоих переменных совпадают, текст блока будет подставлен в результирующий файл.

Функции

Для формирования сложных файлов, требующих вычисления во время обработки, служат функции. Подобно переменным, функции вставляются в текст шаблона при помощи конструкции "#-функция()-#".

Функции, использующие в аргументах путь к файлу (**path**), могут использовать в качестве домашней директории пользователя '~'

Доступные функции:

belong - (**не используется начиная с 3.1.1**) - назначение такое же как у `merge`, за исключением, что она работает только с именем пакета, не учитывая категорию.

merge([category/pkg_name]) - проверка имени устанавливаемого пакета, если выполняется установка пакета. В остальных случаях функция будет возвращать положительный результат.

`category` - категория пакета.

`pkg_name` - имя пакета.

Если у функции нет аргумента, то именем пакета становится имя шаблона (для директорий - имя директории), категорией - родительская директория. Причем если в имени категории или имени пакета есть версия и/или число для сортировки (20-mc,40-xfce-4.6), то они отбрасываются.

Примеры:

sys-apps/15-portage-2.2/.calculate_directory, category=sys-apps,pkg_name=portage sys-fs/udev, (portage файл), category=sys-fs,pkg_name=udev Результат зависит от значения переменной шаблона `cl_merge_pkg` (в версиях ниже 3.1.1 `cl_belong_pkg`)

- значение `cl_merge_pkg` - "
 - результат '1', независимо от того, есть или нет аргумент у функции
- значение `cl_merge_pkg` - 'имя_пакета'
 - если совпадают значения `cl_merge_pkg` и аргумента функции, результат - '1' иначе ''

- если у функции нет аргумента то сравниваются значение cl_merge_pkg и имя директории в которой находится шаблон с функцией merge(), в случае совпадения результат - '1' иначе "

При выполнении шаблонов для модификации исходного кода пакетов (ac_install_patch) функция в случае положительного результата возвращает номер версии вместо 1.

Пример.

Заголовок шаблона1:

```
# Calculate merge(sys-auth/nss_ldap) !=
```

Заголовок шаблона2:

```
# Calculate merge(kde-libs/kdm) !=
```

1. Наложим шаблон для программы sys-auth/nss_ldap: Установим значение переменной cl_merge_pkg равным sys-auth/nss_ldap. Будет применен только шаблон1
2. Наложим все шаблоны. Значение переменной cl_merge_pkg по умолчанию - ". Будут применены шаблон1 и шаблон2.

Если параметром для merge указан (или определен по пути шаблона) не существующий пакет, то функция возвращает пустое значение. **case(type,var)** - вывод значения переменной шаблона с изменением регистра символов.

где:

- *type* - тип изменения регистра: upper - верхний регистр, lower - нижний регистр, capitalize - первая буква в верхнем регистре.
- *var* - название переменной шаблона.

Пример. Выведем название хоста в верхнем регистре:

```
#-case(upper,os_net_hostname)-#
```

disk(mount_point,name) - выводит значение параметра жесткого диска при инсталляции системы.

Значение функция получает из переменной ('os_install_disk_' + name, если такой не существует, то 'os_disk_' + name), для поиска нужного значения используется переменная os_install_disk_mount (точки монтирования при инсталляции). где:

- *_mount_point_* - точка монтирования при инсталляции.
- *name* - последний элемент переменных начинающихся на os_install_disk_/os_disk ('os_instll_disk_/' + name).

значения переменных

```
os_install_disk_mount = ['swap', '/', "", "", '/var/calculate']
os_disk_grub ['0,0', '0,1', '0,2', '0,3', '0,4']
```

функция

disk(/var/calculate,grub)

- в переменной os_install_disk_mount находим /var/calculate, получаем индекс 4
- в переменной os_disk_grub по индексу 4 получаем значение '0,4'
результат функции disk(/var/calculate,grub) '0,4'

функция

disk(/boot,grub)

- в переменной os_install_disk_mount /boot не найден, индекс отсутствует
- если индекс отсутствует, в переменной os_disk_install находим /, получаем индекс 1
- в переменной os_disk_grub по индексу 1 получаем значение '0,1'
результат функции disk(/boot,grub) '0,1'

Пример.

Выведем диск для загрузчика grub:

```
#-disk(/boot,grub)-#
```

получаем значение 0,1

elog(pkg) - получить отметку времени (timestamp) установки указанного пакета. Если пакет не указан, возвращается время последнего установленного пакета. Информация извлекается из `emerge.log`. Функция используется для определения необходимости настройки профиля пользователя при входе его в сеанс.

где:

`pkg` - полное название пакета с категорией

env(service.var_name) - чтение значения записанной переменной шаблона сервиса. Информация получается путем обработки файлов хранения значений переменных шаблонов. где:

- `service` - название сервиса.
 - `_var_name_` - переменная сервиса.
- разделитель - '.' точка Примеры:
- прочитаем доменное имя для соединения с jabber сервисом

```
#-info(jabber,sr_jabber_host)-#
```

получаем значение `jabber.calculate.ru`

exists(path,opt) - проверка существования файла или директории.

Если файл или директория существует, выводит '1', иначе "

`path` - путь в файловой системе.

`opt` - необязательная опция. Возможные значения "opt":

- `root` - путь к файлу не будет содержать chroot-пути. (Какой путь указан, такой будет в действительности - вне зависимости от переменных `cl_chroot_path` и `cl_root_path`)

Пример. Проверим существование директории `/etc`:

```
#-exists(/etc)-#
```

Результатом будет '1'.

grep(file,regex) - *(добавлена в 3.3.1) получить значение из файла `file`, соответствующее регулярному выражению `regex`. Если в регулярном выражении используется группировка значений, функция возвращает содержимое первой группы.

Начиная с версии 3.3.1.2 преобразовывается \xFF, где FF двузначный шестнадцатеричный код символа.

Пример. Получить значение `nofscks` из `dracut.conf`.

```
#-grep(/etc/dracut.conf,nofscks="( .*? )")-#
```

Специальные символы регулярного выражения (РВ):

"." - соответствует любому символу, исключая новую строку. При включённом режиме **dotall** соответствует любому символу.

"^" - соответствует началу файла. При включённом режиме **multiline** соответствует началу строки.

"\$" - соответствует концу файла. При включённом режиме **multiline** соответствует концу строки.

"*" - соответствует 0 и более повторений предшествующего РВ. РВ будет соответствовать максимально возможной части текста.

"*?" - "не жадная" версия использования предыдущего символа. РВ будет соответствовать минимально возможной части текста.

"+" - соответствует 1 и более повторений предшествующего РВ. РВ будет соответствовать максимально возможной части текста.

"+?" - "не жадная" версия использования предыдущего символа. РВ будет соответствовать минимально

возможной части текста.

"?" - соответствует 0 или 1 повторению предшествующего РВ. РВ будет соответствовать максимально возможной части текста.

"??" - "не жадная" версия использования предыдущего символа. РВ будет соответствовать минимально возможной части текста.

"{m,n}" - соответствует от m до n повторений предшествующего РВ.

"{m,n}?" - "не жадная" версия использования предыдущей конструкции. РВ будет соответствовать минимально возможной части текста.

"\\" - экранирование спецсимволов

"[]" - соответствует любому символу из указанного набора.

"[^]" - соответствует любому символу не из указанного набора.

"A|B" - создать РВ соответствующее и A и B.

"(...)" - группировка РВ с сохранением групп (grep возвращает содержимое первой группы)

"(?ims)" - устанавливает режимы РВ: регистронезависимый, multiline, dotall.

"(?:...)" - группировка РВ без сохранения групп

"(?=...)" - проверка текста после РВ на совпадение.

"(?!=...)" - проверка текста после РВ на не совпадение.

"(?<=...)" - проверка текста перед РВ на совпадение (это РВ должно быть фиксированной длины).

"(?<!...)" - проверка текста перед РВ на не совпадение (это РВ должно быть фиксированной длины).

"\w" - соответствует любому символу из набора [a-zA-Z0-9_]

"\W" - соответствует любому символу не из набора [^a-zA-Z0-9_]

"\s" - соответствует любому пробельному символу

"\S" - соответствует любому не пробельному символу

"\d" - соответствует любой десятичной цифре [0-9]

"\D" - соответствует любому символу, не являющемуся десятичной цифрой [^0-9]

groups(group1,group2,..groupN) - проверка вхождения пользователя в группы group1,group2, ..groupN

Если пользователь входит хотя бы в одну из групп выводит '1', иначе "

group1 .. groupN - названия групп.

Пример. Проверим входит ли пользователь в группу wheel:

```
#-groups(wheel) -#
```

Если пользователь входит в группу wheel, результатом будет '1'.

ini(var, value, opt) - запись и чтение переменной из конфигурационного файла пользователя (~/.calculate/ini.env).

Начиная с версии 2.2.20-r4: если функция выполняется для настройки системы, то конфигурационный файл будет находиться в /etc/calculate/ini.env. где:

- var - имя переменной функции. Имя должно начинаться с буквы и может содержать латинские буквы и цифры, а также точку. Точка служит разделителем для раздела и имени переменной для размещения в конфигурационном файле.
- value - значение переменной функции, значение присваивается переменной функции и записывается в конфигурационный файл (функция при этом возвращает пустое значение). При отсутствии второго аргумента переменная считывается из конфигурационного файла; при отсутствии названия переменной в конфигурационном файле в переменную записывается пустое значение.

Начиная с версии 2.2.20-r4, если аргумент пустая строка без кавычек, то переменная удаляется. Если аргумент - пустая строка в кавычках, то значение переменной в ini-файле очищается.

- opt - опция преобразования значения переменной, необязательный параметр; возможные значения url, purl, unicode. При использовании этого аргумента второй аргумент функции должен быть пустым.

Примеры:

1. Создадим переменную "test" и присвоим ей значение 15, запишем в конфигурационный файл:

```
#-ini(test,15)-#
```

2. Считаем значение ранее записанной в конфигурационный файл переменной test, заменим функцию значением считанной переменной, в нашем случае "15":

```
#-ini(test)-#
```

3. Использование функции ini с тремя аргументами. Предположим, что в файле `~/.calculate/ini.env` существует секция

```
[test]
param = Тестовый параметр
```

тогда функция

```
#-ini(test.param,,unicode)-#
```

вернет

```
\u0422\u0435\u0441\u0442\u043e\u0432\u044b\u043f\u043f\u0430\u0440\u0430\u043c\u0435\u0442\u0440
```

а функция

```
#-ini(test.param,,url)-#
```

вернет

```
%d0%a2%d0%b5%d1%81%d1%82%d0%be%d0%b2%d1%8b%d0%b9%20%d0%bf
%d0%b0%d1%80%d0%b0%d0%bc%d0%b5%d1%82%d1%80
```

Опция `purl` отличается от `url` тем, что преобразует '/' в код '%2F'

4. помещение сервиса sshd в автозапуск с проверкой и отметкой в `ini.env`

```
# Calculate path=/etc/runlevels/default name=sshd link=/etc/init.d/sshd
symbolic ini(runlevels.openssh)!=on&&ini(runlevels.openssh,on)==
```

livemenu(mode) - (добавлена в версии 3.4) - функция специального назначения для формирования мультизагрузочного меню для Flash. `mode` - режим работы функции, и может быть следующим:

- `submenu` - возвращает список систем в следующем формате:

идентификатор системы;
полное название в загрузочном меню;
путь до ядра;
параметры для загрузки;
путь до `initrd`;
параметры загрузки `splash`;

Пример:

```
cl-2;
Calculate Linux Desktop Xfce;
/boot/vmlinuz-2;
root=live:LABEL=CALCULATE;
/boot/initrd-2;
splash;
```

- `xorg` - возвращает список систем, в которых установлен `xorg-server`. Например `cl-1 cl-2 cl-3`.

- `video` - возвращает список систем, в которых присутствуют проприетарные драйверы. Например `c1-2 c1-3`

Функция используется в шаблонах `6_ac_builder_iso/0_bootmenu/desktop.config`, а также для формирования `gfxboot.cfg`.

server(service.option,var) - чтение значения параметра сервиса. Информация получается путем обработки файла `/var/calculate/remote/server.env`, где:

- *service* - название сервиса.
- *option* - опция сервиса.
- *var* - имя переменной функции(необязательный параметр).

Разделитель *service* и *option*: '.' - точка.

В случае использования *var* - имени переменной функции, полученное значение опции сервиса будет записана в переменную функции *var*.

Если *var* не используется, полученное значение опции сервиса будет вставлено в текст конфигурационного файла. Примеры:

- прочитаем значение порта для соединения с jabber-сервисом

```
#-server(jabber,port)-#
```

получаем значение 5223

1. прочитаем значение порта для соединения с jabber-сервисом и поместим в переменную функции `jabber_port`

```
#-server(jabber,port,jabber_port)-#
```

Значение переменной функции `jabber_port` - 5223. **list(var,index)** - вывод значения по индексу из переменной.

где:

- *var* - название переменной функции или переменной шаблона (значение переменной должно быть списком).
- *index* - индекс для поиска значения в переменной функции или переменной шаблона (первый элемент имеет индекс 0, и т.д.).

Пример.

Значение переменной

```
os_disk_dev = ['/dev/sda1', '/dev/sda2', '/dev/sda3', '/dev/sda4', '/dev/sda5']
```

```
#-list(os_disk_dev,1)-#
```

Метка функции будет заменена на `'/dev/sda2'`.

Если запрошен элемент, отсутствующий в списке, наложение шаблонов прерывается с ошибкой.

Начиная с версии 2.2.17 - возвращает пустую строку.

in(var,value1,value2...) - проверяет есть ли среди значений переменной *var*, хотя бы одно значение из *value*. Если значение найдено, то возвращает "1", иначе пустую строку. Переменная может быть как списком, так и строкой.

- *var* - название переменной
- *valueX* - список значений для сравнения

Пример:

Значение переменной строка

```
os_install_linux_shortname = CLDX
```

```
#-in(os_install_linux_shortname,CLDX,CLD,CLDG)-#
```

Пример 2:

Значение переменной список
os_linux_pklist = CLDX,base
#-in(os_linux_pklist,CLDX,CLD,CLDG)-#

Вернет 1, так как есть общее значение - CLDX.

Метка функции будет заменена на CLDX.

kernel(kernel_opt) - (добавлена в версии 3.3.1) получить значение параметра конфигурации ядра. Возвращает "y", "m" или пустую строку. y - опция включена в ядро, m - модуль ядра, пустая строка - опция выключена.

_kernel_opt - регистрационезависимое название опции без префикса CONFIG\

Пример 1. Включен ли ext4:

kernel(ext4_fs)!=

Пример 1. reiserfs - модуль ядра, а ext4 - включен в ядро

kernel(reiserfs_fs)==m&&kernel(ext4_fs)==y

load(arg,path,opt) - отображение информации из файла.

где:

arg - тип содержимого файла;

path - путь к файлу;

opt - необязательная опция Возможные значения "opt":

- *root* - путь к файлу не будет содержать chroot-пути. (Какой путь указан, такой и будет в действительности, вне зависимости от переменных cl_chroot_path и cl_root_path.)

Возможные значения "arg":

- *ver* - содержимое файла номер версии
- *num* - содержимое файла число
- *char* - содержимое файла строка
- *empty* - результат, содержимое файла без закомментированных (комментарии # или ;) и пустых строк. Возможные значения "path":
- *path* - абсолютный или относительный путь к файлу.

Пример. Выведем содержимое /proc/cpuinfo на место объявления функции:

#-load(char,/proc/cpuinfo)-#

module(name_space) - (удалена в 3.1) получение переменных шаблонов пакета или выполнение методов пакета, используя его арі-модуль или выполнение определенного метода для всех пакетов, у которых есть арі-модуль,

где:

- *_name_space_* - пространство имен арі-модуля.

Пространство имен для получения переменных состоит из элементов разделенных точкой.

Первый элемент пространства имен - имя пакета. Имя пакета это название установленного пакета, имеющего арі-модуль (тире '-' в названии должно быть заменено на нижнее подчеркивание '_').

Первый элемент пространства имен для всех пакетов - 'all' (зарезервированное название).

Пример первого элемента для пакета calculate-ldap:

calculate_ldap

Второй элемент пространства имен - название метода `api` или '`var`' - зарезервированное название для пространства имен переменных шаблонов пакета.

Пример первого и второго элементов для пакета `calculate-ldap` для получения доступа к переменным:

```
calculate_ldap.var
```

Пример первого и второго элементов для пакета `calculate-ldap` для получения результата выполнения метода `is_setup()` `api`-модуля (проверка, настроен ли пакет).

```
calculate_ldap.is_setup
```

Третий элемент пространства - в большинстве случаев это пространство имен для получения значения переменной шаблона.

Пример пространства имен для получения значения переменной шаблонов `cl_name` пакета `calculate-ldap`:

```
calculate_ldap.var.cl_name
```

Примеры использования функции:

Получим значение переменной шаблонов `cl_name` (название пакета) пакета `calculate-ldap`.

```
#-module(calculate_ldap.var.cl_name)-#
```

Получаем значение

```
calculate-ldap
```

Проверим, настроен ли пакет `calculate-ldap` для работы:

```
#-module(calculate_ldap.is_setup)-#
```

если пакет настроен, получаем значение

```
1
```

Проверим, настроены ли для работы все установленные пакеты, имеющие `api`-модуль:

```
#-module(all.is_setup)-#
```

Если все пакеты настроены, получаем значение

```
1
```

`pkg (category/package)` - вывод версии установленного пакета, если в системе установлено несколько версий этого пакета, ты выводится версия максимального слота.

где:

- *category* - категория пакета
- *package* - название пакета
- Начиная с версии утилит 3.1.0_beta2 добавилась возможность указывать слот пакета через двоеточие: `pkg(kde-base/kdelibs:4)`
- Начиная с версии утилит 3.3 добавлена возможность проверять наличие USE флагов у пакета. Проверяемые USE флаги указываются в квадратных скобках, через запятую. Указание "-" перед USE флагом означает, что флаг у пакета должен быть выключен.

В качестве совместимого режима можно указывать только название пакета; в этом случае скорость обработки шаблона будет ниже.

Пример. Выведем версию установленного пакета (например 4.2.4):

```
#-pkg(kde-base/kdelibs)-#
```

Значит, в системе установлен пакет **kdelibs-4.2.4**.

Пример использования USE. Вывести версию установленного пакета, если он собран с включенным **desktop** и выключенным **client** флагами.

```
#-pkg(sys-apps/calculate-utils[desktop,-client])-#
```

print(message) - вывод на экран информационного сообщения во время выполнения шаблона

warning(message) - вывод на экран предупреждающего сообщения во время выполнения шаблона

error(message) - вывод на экран сообщения об ошибке во время выполнения шаблона

push(var, value) - помещение значения переменной в стек переменных функций шаблонов. где:

- *var* - имя переменной функции. Имя должно начинаться с буквы и может содержать латинские буквы и цифры. Переменная создается на время обработки конфигурационного файла.
- *value* - значение переменной функции, значение присваивается переменной функции и заносится в стек переменных функций шаблонов. При отсутствии второго аргумента, в стек переменных функций шаблонов будет записано значение переменной. Примеры:
- Создадим переменную "test" и присвоим ей значение 15, запишем в стек переменных функций шаблонов.

```
#-push(test,15)-#
```

1. Запишем в стек переменных функций шаблонов значение ранее созданной переменной **test**:

```
#-push(test)-#
```

pop(var) - извлечение значения из стека переменных функций шаблонов и присвоение его переменной.

- *var* - имя переменной функции. Имя должно начинаться с буквы и может содержать латинские буквы и цифры. Переменная создается на время обработки конфигурационного файла. Примеры:
- Создадим переменную "test" и присвоим ей значение 15, запишем в стек переменных функций шаблонов.

```
#-push(test,15)-#
```

1. Получим значение из стека переменных функций шаблонов и присвоим его переменной **test2**:

```
#-pop(test2)-#
```

Получить значение из стека переменных функций шаблонов возможно как в текущем шаблоне, так и в любом другом.

После получения значения оно удаляется из стека. **replace(old, new, var)** - замена в значении переменной **var** текста *old* на *new*.

где:

- *old* - текст, каждое вхождение которого в значении переменной будет заменено на текст *new*
- *new* - текст, на который в значении переменной будет заменен текст *old*
- *var* - название переменной функции или переменной шаблона

Текст *old* и *new* должен быть заключен в двойные или одинарные кавычки.

В тексте в двойных кавычках обрабатываются управляемые символы ('', "", \n, \r, \t, \\).

Начиная с версии 3.2.3-r3 в двойных кавычках также обрабатывается \xFF, где FF двузначный шестнадцатеричный код символа.

В одинарных кавычках текст не обрабатывается.

Пример.

значение переменной шаблона `_ur_signature_`

Компания «Калкулэйт»\nРоссия, Санкт-Петербург, пл. Стажек,
4\nhttp://www.calculate.ru\n+7 812 3363632\n+7 495 7727678

при выполнении функции

```
#-replace('\'n','\'n',ur_signature)-#
```

в точку вставки будет вставлен следующий текст (символ "\n" будет заменен на перевод строки)

Компания «Калкулэйт»
Россия, Санкт-Петербург, пл. Стажек, 4
http://www.calculate.ru
+7 812 3363632
+7 495 7727678

Ниже приведенный пример позволяет преобразовать пробел в значении `hr_board_model` в символ подчеркивания.

```
...
|main.hr_board_model | rs |          |P8H77-V LE|
...
#?replace("\x20","_",main.hr_board_model)==P8H77-V_LE#
...
#replace#
```

`rnd(type,len)` - вывод случайной комбинации символов.

где:

- `type` - используемые символы, возможные значения: num - числа, pas - числа и буквы, uuid - числа и буквы от a-f. (uuid добавлен начиная с версии 2.2.17).
- `len` - количество символов (число).

Пример. Выведем три 3 случайных числа (например 372):

```
#-rnd(num,3)-#
```

`sum(var,sum_print, sum_out)` - вычисляемые значения смещений. Функция разрабатывалась для настройки [Plasma](#) (KDE). где:

- `var` - имя переменной функции "sum". Имя должно начинаться с буквы и может содержать латинские буквы и цифры. Переменная создается на время обработки конфигурационного файла.
- `_sum_print_` - арифметическое выражение, результат которого будет отображен на месте объявления функции. При отсутствии аргумента значение выводиться не будет. Поддерживаются операции сложения "+" и вычитания "-", деления "/" и умножения "*". В арифметическом выражении могут участвовать другие переменные функции, а также переменные шаблона.
- `_sum_out_` - арифметическое выражение, результат которого будет присвоен переменной "var" (первому аргументу функции). При отсутствии третьего аргумента переменной "var" будет присвоено вычисленное значение второго аргумента. Примеры:
- Создадим переменную "clock" и присвоим ей значение 15, не выводя результат.

```
#-sum(clock,,15)-#
```

1. Создадим переменную "bt", присвоим ей значение переменной "clock" и выведем значение.

```
#-sum(bt,clock)-#
```

1. Разместим кнопку на панели шириной "35" пикселов, оставив отступ в "4" пикселя по краям:

```
#-sum(bt, bt+2, bt+35+2)-#
```

wallpaper(resolution, wallpapers_path) - (добавлена в 3.2.3) функция выбирает наиболее подходящее по указанному разрешению изображение из папки. где:

- *resolution* - требуемое разрешение (текущее разрешение можно получить из переменных `os_x11_resolution`, `os_install_x11_resolution`)
- *_wallpapers_path_* - путь, где находится изображение в разных разрешениях. Формат файлов: `разрешение.расширение`. Например: `1024x768.jpg`

Пример: Папка `/usr/share/wallpapers/1` содержит следующие изображения:

- `1024x768.jpg`
- `1280x1024.jpg`
- `1680x1050.jpg`
- `1920x1080.jpg`

```
#-wallpaper(1280x720,/usr/share/wallpapers/1)-#
```

вернет `1920x1080.jpg` как наиболее подходящую по пропорциям.

Для получения текущего разрешения можно использовать выше указанные переменные

```
#-wallpaper(#-install.os_install_x11_resolution-  
#, /usr/share/wallpapers/1)-#
```

Способы объединения

Существуют несколько способов объединения шаблона установки с исходным файлом системы:

- **join** - основной способ объединения - методом слияния двух файлов. Подробно описан в "схеме объединения".
- **before** - шаблон переписывается в начало оригинального файла
- **after** - шаблон переписывается в конец оригинального файла
- **replace** - шаблон переписывается заменяя оригиналный файл
- **delete** - объединение не происходит, оригиналный файл удаляется
- **remove** - вместо объединения происходит удаление оригинального файла Изменение шаблона перед объединением, для способов объединения "before", "after", "replace":
 - если существуют *управляющие элементы*, то они обрабатываются
 - если существует *заголовок*, то он обрабатывается
 - если существует параметр *format* в заголовке то происходит обработка *управляющих символов*: "+", "-", "!"

Правила по умолчанию

По умолчанию способ объединения устанавливается в соответствии с форматом файла.

Для форматов файлов основных приложений "samba", "bind" и т.п. по умолчанию действует объединение "join", для формата "raw" и "bin" - объединение "replace". Для пустого файла по умолчанию действует правило объединения "delete" - конечный файл удаляется из системы.

Правила объединения могут быть изменены установкой параметра "append" заголовка файла шаблона.

Расположение

Правила именования файлов

Имена файлов шаблона могут содержать условные операторы, определяющие правила, при выполнении которых файл шаблона будет перенесен в систему. Подобно ссылке на страницу сайта условия отделяются от имени знаком вопроса (?), после которого могут идти условные операторы, подробно описанные в условных блоках (см. выше).

Арифметические операции могут объединяться условием И (&) и ИЛИ (?). Приоритет в данном случае будет отдаваться условию И.

Пример:

```
grub.conf?os_linux_shortname==CDS?os_linux_shortname==CLD
```

В приведенном примере файл шаблона выполнит настройки загрузчика как для систем [Calculate Linux Desktop](#), так и для [Calculate Directory Server](#).

При наличии условных операторов в заголовке файла шаблона для переноса файла шаблона в систему должны выполняться оба условных выражения.

Правила именования директорий

Правила именования директорий схожи с правилами именования файлов. При выполнении условий, заданных условными операторами, директория будет перенесена в систему; в противном случае директория вместе с содержимым будет пропущена.

Права доступа

Права доступа конфигурационного файла после объединения с файлом шаблона устанавливаются по следующим приоритетам:

- при наличии конфигурационного файла в системе права будут сохранены неизменными;
- в случае отсутствия оригинального конфигурационного файла, права будут выставлены как 644 для файла и 755 для директории;
- в случае установки значения переменной "chmod" или "chown" заголовка файла шаблона, права на конфигурационный файл будут изменены согласно установленному значению.

Символические ссылки

Для создания символьических ссылок используйте параметры "link + symbolic" заголовка файла шаблона.

Если помимо заголовка файл шаблона будет содержать тело шаблона, оригинальный файл будет модифицирован согласно правилам объединения.

Calculate не будет переносить так называемые "битые ссылки" - ссылки, не ведущие ни на какой файл (директорию), если явно не указан параметр **force**.

Схема объединения

Объединение - изменение настроек оригинального файла настроек в соответствии с настройками *файла шаблона*.

В процессе объединения все записи оригинального файла и файла шаблона разбиваются на элементы: *области, переменные, списки, разделённые списки, комментарии, управляющие элементы* (см. ниже).

Файл шаблона должен быть составлен с применением синтаксиса оригинального файла. Расположение элементов оригинального файла при объединении сохраняется. Во время объединения комментарии из файла шаблона не переносятся.

Правила объединения

Над элементами могут происходить операции *Объединение, Замена, Удаление*:

Объединение

- Отсутствующие в оригинальном файле элементы дописываются в конец области. При этом в случае наличия перевода строки перед вставляемым элементом перевод строки добавляется после вставляемого элемента. В противном случае перевод строки добавляется перед вставляемым элементом.
- В случае объединения разделённого списка, отсутствующие элементы добавляются следом за последним элементом разделённого списка конфигурационного файла.

Замена

- Значение элементов заменяется на новое. При этом форматирование переносится из файла шаблона.

Удаление

- Элемент удаляется вместе с переводом строки, стоящим перед элементом.

Правила объединения, действующие по умолчанию

Правила объединения действуют на **элементы с одним именем**, расположенные в **одной области шаблона**. При нахождении различий, по умолчанию действуют следующие правила:

- **Области** - содержимое двух областей объединяется (+).
- **Переменные** - значения переменных заменяются (-).
- **Списки** - содержимое списков заменяется (-).
- **Разделённые списки** - аналогично правилу объединения переменных - значения списков заменяются (-).

Изменение правил объединения

Для изменения правил объединения действующих по умолчанию, в начале **имени элемента** в файле шаблона добавляется управляющие символы:

- "+" - объединить элементы (для областей и списков); после объединения остаются только уникальные элементы
- "-" - значение элемента заменяется
- "!" - элемент удаляется

В CXmlConf описания файла шаблона эти правила описываются тэгом "".

Формат CXmlConf

CXmlConf - универсальный формат описания конфигурационных файлов. Служит для **выборочного изменения настроек** большинства распространенных типов конфигурационных файлов ОС Linux/Unix.

XML файл описания настроек разбивает конфигурационный файл на логические структуры - **элементы**, пригодные для последующего объединения. После объединения файл может быть преобразован в первозданный вид за некоторыми исключениями (см. Схема объединения).

Описания элементов вкладываются в конструкцию:

```
<cxmlconf>
  <head>
    <ver>версия формата</ver>
    <format>формат конфигурационного файла</format>
  </head>
  <body>
    [<area>...</area>... <field>..</field>]
  </body>
</cxmlconf>
```

Где:

- *ver* - передает номер версии разметки
- *format* - формат конфигурационного файла (определяется по распространенным программам).

Все элементы (см. ниже) помещаются внутрь элемента .

Области

Области конфигурационных файлов разграничивают **пространство имен переменных**. Области могут содержать логические структуры, в том числе другие области (пример: {{Filename|named.conf}}).

Области помещается в конструкцию:

```
<area>
  <caption>
    <name>Заголовок области</name>
    <action>join|replace|drop</action>
    <quote>Начальная часть области (заголовок)</quote>
    <quote>Завершающая часть описания области</quote>
  </caption>

  [<field></field>...]

</area>
```

Переменные

Переменные имеют запись в виде:

```
<field type="var">
  <name>имя переменной</name>
  <value>значение переменной</value>
  <action>join|replace|drop</action>
  <quote>Оригинальный текст описания</quote>
</field>
```

В некоторых конфигурационных файлах, например, /etc/openldap/slapd.conf, встречается конструкция:

```
index  cn          pres,sub,eq
index  sn          pres,sub,eq
index  uid         pres,sub,eq
```

В этом случае имя переменной состоит из первой и второй части, а значение - из третьей.

Строка

```
index  cn          pres,sub,eq
```

В XML это будет выглядеть так:

```
<field type="var">
  <name>indexcn</name>
  <value>pres,sub,eq</value>
  <quote>index  cn          pres,sub,eq</quote>
</field>
<field type="br" \>
```

Списки

По примеру файла named . conf, блок "listen-on" может содержать одни значения - значения блока, а не переменных.

Для обозначения значений служит конструкция:

```
<field type="list">
  <name>hostsallow</name>
  <value>192.168.0.0/24</value>
  <value>127.</value>
  <action>join|replace|drop</action>
  <quote>Оригинальный текст списка</quote>
</field>
```

где внутри блока "" сохраняется оригинальный текст описания значения, без завершающего перевода строки.

Разделённые списки

Файл настроек веб-сервера *Apache* может содержать инструкцию "Include", позволяющую делать исходный файл модульным. Подобные случаи описываются в "CXmlConf", как "Разделённые списки".

Разделенные списки описываются следующей конструкцией:

```
<field type="seplist">
  <name>Include</name>
  <value>/etc/apache2/modules.d/*.conf</value>
  <action>join|replace|drop</action>
  <quote>Оригинальный текст списка</quote>
</field>
```

Комментарии

При объединении конфигурационных файлов комментарии оригинального файла сохраняются в неизменном виде.

Все типы комментариев помещаются в конструкцию "". В тексте, помещаемом в "", сохраняются символы комментария и перевод строки:

```
<field type="comment">
  <quote>Оригинальный текст комментария</quote>
</field>
```

Комментарии не могут быть вложенными (быть описаны в других конструкциях комментариев).

Управляющие элементы

Для обозначения **перевода строк** служит конструкция:

```
<field type="br">
  <quote>разделительные элементы (пробелы, табуляция)</quote>
</field>
```

Где:

- "quote" содержит элементы форматирования (пробелы, табуляция)

Формат XML

Поддерживаемые форматы XML

В настоящее время реализована поддержка форматов xml_xfce (XML файл для конфигурирования оконного менеджера XFCE), xml_xfcepanel (XML файл для конфигурирования панелей оконного менеджера XFCE), xml_gconf (XML файл для конфигурирования GNOME)

Отличия от формата CXmlConf

- XML файлы хранятся и обрабатываются без перевода их в другой формат.
- Все правила, функции, переменные действуют на файл XML-шаблона аналогично обычному шаблону за исключением операций объединения элементов XML

Объединение элементов XML

Для объединения элементов XML в тексте XML шаблона используется атрибут XML-элемента - 'action'
Допустимые значения атрибута

- action="join" - элементы объединяются (действует по умолчанию)
- action="replace" - элемент шаблона замещает элемент файла
- action="drop" - элемент файла удаляется

Формат xml_xfce

Формат состоит из элементов "channel" и "property".

В атрибутах "channel" находится имя и версия файла.

Внутри элемента "channel" находятся элементы "property".

Пример элемента "channel":

```
<channel name="xfwm4" version="1.0">
  <property name="general" type="empty">
    ....
  </property>
</channel>
```

Пример элемента "property":

```
<property name="wrap_workspaces" type="bool" value="false"/>
```

У каждого элемента "property" есть атрибут type.

Если type="array", то этот элемент и внутренние элементы будут заменены на элементы из шаблона.
Иначе элементы будут объединены.

Пример type="array":

```
<property name="workspace_names" type="array">
  <value type="string" value="Рабочее место 1"/>
  <value type="string" value="Рабочее место 2"/>
  <value type="string" value="Рабочее место 3"/>
  <value type="string" value="Рабочее место 4"/>
</property>
```

Формат xml_xfcepanel

Формат состоит из следующих элементов: "panels", "panel", "properties", "property", "items", "item".

Если элемент "items", то этот элемент и внутренние элементы будут заменены на элементы из шаблона.
Иначе элементы будут объединены.

Пример элемента "items":

```
<items>
  <item name="xfce4-mixer-plugin" id="9"/>
  <item name="clock" id="10"/>
  <item name="separator" id="52"/>
  <item name="actions" id="12"/>
</items>
```

Формат xml_gconf

Формат состоит из элементов "entry".

Если у элемента "entry" есть атрибут "ltype" (список) или атрибут *type="string"* то этот элемент и внутренние элементы будут заменены на элементы из шаблона, в ином случае элементы будут объединены.

Атрибут элемента "entry" - "mtime" при изменении элемента "entry" показывает время, когда произошло изменение в секундах.

Пример шаблона xml_gconf:

```
# Calculate format=xml_gconf
<?xml version="1.0"?>
<gconf>
  <entry name="rgba_order" mtime="1235158855" type="string">
    <stringvalue>rgb</stringvalue>
  </entry>
  <entry name="dpi" mtime="1235162438" type="float" value="86">
  </entry>
  <entry name="hinting" mtime="1235266915" type="string">
    <stringvalue>full</stringvalue>
  </entry>
  <entry name="antialiasing" mtime="1235266915" type="string">
    <stringvalue>rgba</stringvalue>
  </entry>
</gconf>
```

Формат xml_gconf_tree

Формат состоит из элементов "dir" и "entry" и их атрибутов.

Атрибуты элементов будут объединены в случае 'join'-объединения.

При объединении шаблона и конфигурационного файла элементы "entry" конфигурационного файла будут заменены, соответствующими элементами "entry" шаблона.

Атрибут элемента "entry" - "mtime" при изменении элемента "entry" показывает время когда произошло изменение в секундах.

Пример шаблона xml_gconf:

```
# Calculate format=xml_gconf_tree
<?xml version="1.0"?>
<gconf>
  <dir name="desktop">
    <dir name="gnome">
      <dir name="volume_manager">
        <entry name="percent_used" mtime="1285580987"
schema="/schemas/desktop/gnome/volume_manager/percent_used"/>
      </dir>
    </dir>
  </dir>
</gconf>
```

```
</dir>
</dir>
<gconf>
```

Примеры использования

Файл шаблона:

```
# Calculate format=xml_xfce
<?xml version="1.0" encoding="UTF-8"?>
<channel name="xfce4-session" version="1.0">
  <property name="general" type="empty" action="drop">
    <property name="FailsafeSessionName" type="empty"/>
    <property name="SessionName" type="string" value="Default"/>
    <property name="SaveOnExit" type="bool" value="true"/>
  </property>
</channel>
```

Строка говорит о том, что этот элемент и все элементы внутри будут удалены.

Получившийся в результате файл имеет следующий вид:

```
<?xml version="1.0" encoding="UTF-8"?>
<channel name="xfce4-session" version="1.0">
</channel>
```

Формат 'patch'

Формат *patch* используется для обработки конфигурационных файлов при помощи регулярных выражений языка программирования Python.

Особенности.

Формат *patch* использует обработку конфигурационного файла на основании шаблона (тип объединения *patch*); при этом не происходит объединения шаблона и конфигурационного файла.

Обработка исходного файла происходит целиком, а не построчно, поэтому символы '^' и '\$' означают соответственно начало и конец файла. Символ '.' означает любой символ исключая перевод строки.

Начиная с версии **calculate-lib-3.1.7-r5** в параметры заголовка для **format=patch** добавлены опции **multiline** и **dotall**. При включенном **multiline** '^' и '\$' обозначают начало и конец строки. При включенном **dotall** '.' включает в себя перевод строки.

Описание.

Шаблон формата *patch*:

```
# Calculate format=patch
<reg>регулярное выражение python 1</reg>
<text>текст 1 для замены регулярного выражения</text>
<reg>регулярное выражение python 2</reg>
<text>текст 2 для замены регулярного выражения</text>
...
```

В содержимом тегов **reg** и **text** символы "&", "

Пример:

```
# Calculate format=patch
<reg>TEXT&DATA</reg>
```

```
<text>TEXT_CONFIG</text>
```

Этот шаблон заменит в конфигурационном файле *TEXT&DATA* на *_TEXT_CONFIG_*.

Формат 'diff'

Добавлен начиная с версии 3.1.4

Формат *diff* используется для наложение *diff* патчей на исходный код пакета. Сам файл в системе не создается, а полученное содержимое обрабатывается относительно каталога указанного в параметре *path*.

Пример:

```
# Calculate format=diff
--- panel-plugin/xkb-cairo.c      2012-07-17 16:23:24.997030066 +0400
+++ panel-plugin/xkb-cairo.c      2012-07-17 16:47:34.107054590 +0400
@@ -27,7 +27,7 @@
 #include "xkb-util.h"
 #include "xfce4-xkb-plugin.h"

#define XKB_PREFERRED_FONT "Courier New, Courier 10 Pitch, Monospace Bold
%d"
#define XKB_PREFERRED_FONT "Droid Sans, Courier New, Courier 10 Pitch,
Monospace Bold %d"
```

Формат 'kernel'

Добавлен начиная с версии 3.3.0.16

Формат *kernel* используется для объединения конфигурационных файлов ядра без сохранения комментариев.

Пример:

```
# Calculate format=kernel
CONFIG_XFS_FS=m
CONFIG_REISERFS_FS=y
# CONFIG_EXT3_FS is not set
!CONFIG_EXT3_FS_POSIX_ACL=
!CONFIG_EXT3_FS_SECURITY=
!CONFIG_EXT3_FS_XATTR=
```

Формат 'dconf'

Добавлен, начиная с версии 3.3.2.9

Формат *dconf* используется для модификации настроек пользователя, хранящихся в реестре *dconf*. Формат *dconf* поддерживает параметр заголовка *dconfpath* для возможности изменять базовый путь настроек.

Пример:

```
# Calculate format=dconf dconf=/org/gnome/eog/
[view]
background-color='#000000'
use-background-color=true

[ui]
statusbar=true
toolbar=true
```

```
sidebar=false
```

Для применения шаблонов формата *dconf*, утилиты используют пакет *gnome-base/dconf*.

Формат 'json'

Добавлен, начиная с версии 3.4.1

Формат *json* используется для модификации настроек в формате json (например *Preferences* пакета *www-client/chromium*).

Пример *json*

```
{"google": {"services": {"signin": {"LSID": "", "SID": ""}}},  
 "intl": {"accept_languages": "ru-RU,ru,en-US,en", "status": 1}}
```

При объединении элементов в конфигурационном файле меняются только конкретные значения.

Для изменения правил объединения действующих по умолчанию, в начале **имени элемента** в файле шаблона добавляется управляющие символы:

- "-" - ветка заменяется
- "!" - ветка удаляется

Пример объединения

```
# Calculate format=json  
{"intl": {"accept_languages": "en"}}
```

```
{"google": {"services": {"signin": {"LSID": "", "SID": ""}}},  
 "intl": {"accept_languages": "en", "status": 1}}
```

Пример удаления

```
# Calculate format=json  
{"!intl": ""}
```

```
{"google": {"services": {"signin": {"LSID": "", "SID": ""}}}}
```

Пример замещения

```
# Calculate format=json  
{"-intl": {"accept_languages": "en"}}
```

```
{"google": {"services": {"signin": {"LSID": "", "SID": ""}}},  
 "intl": {"accept_languages": "en"}}
```

Ветка настроек также будет замещена если в шаблоне содержится значение.

Пример замещения

```
# Calculate format=json  
{"intl": 5}
```

```
{"google": {"services": {"signin": {"LSID": "", "SID": ""}}},  
"intl": 5}
```

Переменные шаблонов

- [Переменные шаблонов](#)
- [Введение](#)
- [Типы переменных](#)
- [Переменные выполняемых действий](#)
- [События](#)
- [Значения переменных](#)
- [Изменение значений переменных](#)
- [Изменение значения переменной в командной строке](#)
- [Изменение значения через конфигурационный файл](#)
- [Использование переменных](#)
- [Вставка значений в шаблон](#)
- [Передача значения функции](#)
- [Условия в заголовках шаблонов](#)
- [Условные выражения](#)

Введение

Утилиты Calculate содержат *переменные*, которые могут быть использованы в [шаблонах](#). Каждый пакет утилит добавляет свой набор переменных. Несмотря на то что переменные принадлежат разным пакетам, их имена уникальны. Для просмотра переменных и их значений используется утилита `cl-core-variables-show`:

```
# cl-core-variables-show  
* Список переменных  
+-----+-----+-----+  
+-----+-----+-----+  
|Переменная| Режим |Расположение |Значение  
+-----+-----+-----+  
+-----+-----+-----+  
|main.ac_custom_name| wc | ''  
|main.cl_action| rs | ''  
|main.cl_autoupdate_set| wb | system | off  
|...  
|install.ac_install_configure| rs | off  
|install.ac_install_disk| rs | off  
|install.ac_install_flash| rs | off  
|...  
|update.ac_update_sync| rs | ''
```

update.cl_rebuild_world_set	wb	off
update.cl_update_rev_set	wb	off
update.cl_update_world	ws	''

Здесь видно, что утилита, отобразила переменные всех пакетов. Колонка "Переменная" отображает модуль переменной и имя переменной, разделенных точкой. "Режим" состоит из двух букв:

- доступ
 - "g" только для чтения
 - "w" поддерживает изменение
- тип переменной
 - "s" строка
 - "b" логическая переменная (on/off)
 - "l" список
 - "t" таблица
 - "c" строка с выбором

"Расположение" указано для тех переменных, чьи значения изменяются в calculate.env файлах. Есть три места в которых можно изменять значения по умолчанию для writeable переменных: system (/etc/calculate/calculate.env), local (/var/calculate/calculate.env), remote (/var/calculate/remote/calculate.env).

"Значение" отображает содержимое переменной. Содержимое переменных не точное, так как при различных командах содержимое может изменяться (например изменение значений при указании опций команды). С помощью параметра --filter можно задать фильтрацию выводимых переменных:

- userset - переменные установленные в calculate.env файлах
- writable - переменные доступные для изменения
- system,local,remote - переменные указанные в конкретном calculate.env файле
- часть имени переменной

Типы переменных

Для удобства в именах переменных используется обозначение типа переменной, часто используется название пакета и может использоваться тип возвращаемого значения.

Всего существует семь типов переменных:

1. ac - переменные выполняемых действий
2. cl - общие настройки утилит
3. hr - настройки оборудования
4. ld - атрибуты LDAP
5. os - операционная система
6. sr - настройки сервисов
7. ur - информация о пользователе

Для примера: переменная cl_install_autoupdate_set относится к общим настройкам утилит, принадлежит пакету calculate-install и содержит одно из двух значений - on либо off.

Переменные выполняемых действий

Пакеты утилит содержат шаблоны настройки, сгруппированные по определенным действиям. Например, настройка профиля пользователя, настройка пакета при установке и т.д. Для того, чтобы определить, какие шаблоны следует накладывать, пакеты утилит содержат специальные переменные выполняемых действий.

calculate-install

- `ac_install_configure` - выполнение шаблонов настройки компонентов системы
- `ac_install_disk` - настройки системы для загрузки с жесткого диска
- `ac_install_flash` - настройка системы для загрузки с USB-Flash
- `ac_install_live` - настройка пакета (динамические параметры)
- `ac_install_merge` - настройка пакета (статические параметры)
- `ac_install_patch` - модификация исходного кода пакета
- `ac_install_pxe` - настройка PXE загрузки
- `ac_install_unmerge` - настройка системы при удалении пакета

calculate-desktop

- `ac_desktop_profile` - настройка пакета в профиле пользователя
- `ac_desktop_merge` - настройка пакета в рабочей системе

calculate-client

- `ac_client_merge` - настройка пакета в рабочей системе
- `ac_client_domain` - настройка машины для работы в домене
- `ac_client_undomain` - настройка машины для локальной работы

calculate-core

- `ac_custom_name` - выполнение произвольного действия **calculate-builder** (**Утилиты Calculate 2.2**)
- `ac_builder_iso` - настройка ISO-образа
- `ac_builder_squash` - настройка Squash-образа **calculate-assemble** (**Утилиты Calculate 2.2**)
- `ac_assemble_prepare` - первичная настройка собираемой системы
- `ac_assemble_setup` - настройка системы во время сборки

События

В зависимости от событий (например, установка пакета), утилиты выставляют значения переменных действий и накладывают шаблоны.

Ниже перечислены события, а также значения переменных действий, записанные через знак двойного равенства, принятый в условных выражениях шаблонов:

Настройка системы

Выполняется во время первой загрузки системы, загрузке с LiveCD, USB-Flash или USB-HDD, при выполнении команды `cl-setup-system --live`.

- `ac_client_domain==on` (если машина настроена на работу в домене)
- `ac_client_undomain==on` (если машина настроена как локальная)
- `ac_client_merge==on`
- `ac_install_live==on`
- `ac_desktop_merge==on`

Установка системы на жесткий диск

Выполняется при установке системы командой `cl-install`.

- `ac_install_merge==on`
- `ac_install_live==on`
- `ac_install_disk==on`

Установка системы на USB-Flash

Выполняется при установке системы, если в качестве носителя используется USB-Flash cl-install.

- ac_install_flash==on

Установка системы для загрузки по сети (PXE)

Выполняется при установке системы на сервере для загрузки по сети cl-install --pxe.

- ac_install_pxe==on

Установка пакета

Выполняется во время установки программы emerge <название_пакета> в builder или через calculate-assemble

- ac_install_merge==on Выполняется во время удаления программы emerge -C <название_пакета> в builder или через calculate-assemble
- ac_install_merge==on
- ac_install_unmerge==on Выполняется во время установки программы emerge <название_пакета> в рабочей системе
- ac_install_merge==on
- ac_install_live==on
- ac_desktop_merge==on
- ac_client_merge==on
- ac_client_domain==on (если машина настроена на работу в домене)
- ac_client_undomain==on (если машина настроена как локальная) Выполняется во время удаления программы emerge -C <название_пакета> в рабочей системе
- ac_install_unmerge==on
- ac_install_merge==on
- ac_install_live==on
- ac_desktop_merge==on
- ac_client_merge==on
- ac_client_domain==on (если машина настроена на работу в домене)
- ac_client_undomain==on (если машина настроена как локальная)

Настройка профиля пользователя

Выполняется при запуске cl-desktop <логин_пользователя> во время входа в сеанс.

- ac_desktop_profile==up Выполняется во время установки программы в рабочей системе emerge <название_пакета>
- ac_desktop_profile==on

Ввод компьютера в домен

Выполняется при вводе компьютера в домен командой cl-client .

- ac_client_domain==on
- ac_client_merge==on

Вывод компьютера из домена

Выполняется при выводе компьютера из домена командой cl-client -r.

- ac_client_undomain==on
- ac_client_merge==on

Подключение удаленных ресурсов домена при загрузке

Выполняется при запуске демона `init.d/client` во время загрузки (`cl-client --mount`)

- `ac_client_merge==on`
- `ac_client_domain==on` (если машина настроена на работу в домене)
- `ac_client_undomain==on` (если машина настроена как локальная)

Обновление портежей

Выполняется после обновления портежей и оверлея командой `eix-sync`

- `ac_update_sync==on`

Модификация исходного кода пакета

Выполняется во время сборки пакета `emerge <название_пакета>` перед его компиляцией

- `ac_install_patch==on`

Подготовка squash-образа (Calculate утилиты 2.2)

Выполняется перед упаковкой в Squash-образ `cl-image iso` или `cl-image squash`.

- `ac_builder_squash==up`

Подготовка ISO-образа (Calculate утилиты 2.2)

Выполняется перед созданием ISO-образа командой `cl-image iso`.

- `ac_builder_iso==up`

Подготовка к сборке дистрибутива (Calculate утилиты 2.2)

Выполняется во время подготовки к сборке дистрибутива, после распаковки Stage-образа, командой `cl-assemble`.

- `ac_assemble_prepare==up`

Настройка системы во время сборки (Calculate утилиты 2.2)

Выполняется во время подготовки к сборке дистрибутива, после добавления оверлея `calculate`, командой `cl-assemble`.

- `ac_assemble_setup==up`

Обновление системы во время сборки (Calculate утилиты 2.2)

Выполняется перед сборкой пакетов в собираемой системе, командой `cl-make -u` или `cl-make -U`.

- `ac_assemble_prepare==up`
- `ac_assemble_setup==up`

Произвольное событие

Выполняется при вызове команды `cl-core-custom имя_события`

- `ac_core_custom==имя_события`

Значения переменных

Переменные могут содержать несколько типов значений:

1. **значение для подстановки** используется для подстановки значения в шаблон;
Пример:

main.os_locale_locale	rs	ru_RU.UTF-8
main.os_net_allow	rs	10.0.0.0/24
main.os_net_ip	rs	10.0.0.84
main.os_x11_video_drv	rs	radeon

1. **значение для условного выражения** участвует в условных выражениях;

Пример:

main.hr_video	rs	ati
install.os_install_linux_system	rs	desktop
install.os_install_locale_language	rs	ru

1. **информация** отображает текущие настройки системы;

Пример:

install.os_net_interfaces_info	rs	enp1s0 (10.0.0.84)
main.hr_video_name	rs	Advanced Radeon HD
7540D		

1. **массив значений** используется другими переменными или функциями.

Пример:

install.cl_migrate_user	wl	guest
install.os_device_data	rt	
/dev/sda,dos,hdd,0,ATA OCZ-VERTEX3,60022480896		
desktop.cl_desktop_online_data	rt	guest,0,6

Переменные разбиты на типы условно. Одно и то же значение может соответствовать нескольким типам.

Изменение значений переменных

При обращении к переменной ее значение определяется программой на основании настроек системы, а также значений других переменных. Значения некоторых переменных можно изменить. Такие переменные отмечены первой буквой *W* в поле "Режим". Остальные переменные, доступные только для чтения, отмечены первой буквой *r*. Пример:

main.cl_ver	rs	3.1.7
install.os_install_x11_composite	wb	on

Изменить значение переменной можно либо из командной строки параметрами утилиты, либо сохранив значение в конфигурационном файле. Если переменная меняется в обоих местах, приоритет отдается командной строке.

Изменение значения переменной в командной строке

Для изменения значений переменных используются различные параметры команд: Например

- для изменения cl_autopartition_root_size в команде cl-install используется опция --root-size
- для изменения os_install_x11_video_drv в команде cl-install используется опция --video

Изменение значения через конфигурационный файл

Вы можете предопределить значения переменных в файле calculate.env. Пути к файлам находятся в переменной cl_env_path:

```
|main.cl_env_path          | wl  |
|/etc/calculate/calculate.env,   |      |
|                                |      |
|/var/calculate/calculate.env,   |      |
|                                |      |
|/var/calculate/remote/calculate.env |
```

Приоритет распределяется от первой к последней записи. Т.е. переменная, измененная в файле /var/calculate/remote/calculate.env, перепишет другие значения.

Пример содержимого файла calculate.env:

```
[install]
os_install_ntp = ntp0.zenon.net
```

Обратите внимание: переменная находится в секции [install], т.е. это модуль calculate-install.

Для изменения значений переменных в calculate.env файлах можно воспользоваться утилитой cl-core-variables с параметром --set. Преимущество этого способа заключается в том, что перед записью значения переменной выполняется проверка на допустимость.

Изменение значения переменной install.cl_autologin

```
cl-core-variables --set install.cl_autologin=guest
```

Изменение значения переменной install.cl_autologin в /var/calculate/calculate.env

```
cl-core-variables --set install.cl_autologin=guest:local
```

Удаление значения install.cl_autologin из calculate.env

```
cl-core-variables --set install.cl_autologin
```

Использование переменных

Переменные - основа шаблонов. Их значения настраивают систему в зависимости от текущего состояния оборудования и определяют логику работы. Здесь мы рассмотрим все случаи использования переменных.

Вставка значений в шаблон

Для вставки значения переменной в шаблон используйте конструкцию #-_имя_переменной_-# или расширенная конструкция #-_модуль_.имя_переменной_-#.

Пример настройки Xorg-сервера:

```
Modes "#-install.os_install_x11_resolution-#"
```

Необходимость указания модуля зависит от значения модуля по умолчанию. Модуль по умолчанию задается параметром env в заголовке шаблона и распространяется на все вложенные шаблоны.

Пример:

```
# Calculate env=install  
Modes "#-os_install_x11_resolution-#"
```

Передача значения функции

Значение переменной, заключенной в #--#, вычисляется и подставляется в шаблон в первую очередь. Если переменная будет записана в параметрах функции, для вычисления последней будет использоваться рассчитанное значение переменной.

Пример:

```
<entry name="paned_size" type="int" value="#-sum(ysize,, #-os_x11_height-  
# / 3)-#" />
```

В примере функция *sum* получит три значения: параметр *ysize*, пустое значение и формулу для расчёта содержимого переменной *os_x11_height* (разрешение экрана по вертикали), поделенное на 3.

Условия в заголовках шаблонов

Заголовок файла шаблона может включать условные выражения, в случае успешного выполнения которых шаблон будет использоваться либо нет при настройке системы.

Пример заголовка шаблона с условием проверки системы:

```
# Calculate os_install_linux_system==server
```

Условные выражения

В настоящее время поддерживаются простые конструкции условных блоков. Условное выражение, состоящее из одной или более переменных (или функций), начинается с #? и заканчивается #. Далее следует блок текста, после чего завершающая конструкция из первой переменной условия, заключенной в #. Пример:

```
#?hr_laptop==#  
numlock on  
#hr_laptop#
```

или

```
#?os_locale_language==ru&&pkg(media-gfx/cldx-themes)!=#  
current_theme calculate_ru  
#os_locale_language#
```

Во втором примере в условном выражении участвует функция *pkg()*, и её значение сравнивается с пустотой. Условие будет выполнено, если в системе используется русский язык и установлен пакет *media-gfx/cldx-themes*.

Хранение настроек профиля пользователя

- [Хранение настроек профиля пользователя](#)
- [~/.calculate/ini.env](#)
- [~/.calculate/desktop.env](#)

- [~/.calculate/server.env](#)
- [~/.logout](#)

~/.calculate/ini.env

Файл предназначен для хранения переменных функции `ini()`. Формат файла - samba.

Пример:

```
[main]
var1 = test VAR 1
```

Примечание:

- Возможно любое название секции (по умолчанию `main`);
- Возможно любое название переменной.

~/.calculate/desktop.env

Файл предназначен для хранения параметров клиента. Формат файла - samba.

Пример:

```
[rsync]
files = <количество_файлов_в_профиле_пользователя>
exitcode = <код_возврата_rsync_при_ошибке>

[main]
status = success
version = 2.1.11
```

Секции:

- `rsync` - секция относящаяся к rsync
- `main` - секция общих параметров Параметры:
- `status` - состояние (`error`, `process`, `success`)
- `version` - версия клиента

~/.calculate/server.env

Файл предназначен для передачи параметров серверу. Формат файла - plasma.

Пример изменения пароля для пользователя на сервере (сервисы unix, samba):

```
[command] [passwd_samba]
run=on
unix_hash=<ssha_хеш_нового_пароля_unix>
samba_lm_hash=<lm_хеш_нового_пароля_samba>
samba_nt_hash=<nt_хеш_нового_пароля_samba>
samba_nt_hash_old=<nt_хеш_старого_пароля_samba>
status=process
date=YYYY-mm-dd_HH:MM:SS
```

Выполнение команды на сервере для создание инкрементального архива:

```
[command] [pack_profile]
run=on
status=process
date=YYYY-mm-dd_HH:MM:SS
```

- action - команда создания архива;
- status - состояние выполнения команды (error, process, success).

~/.logout

Устаревший формат файла. Файл предназначен для хранения состояния клиента (совместимость со старыми версиями сервера и клиента). Возможное содержимое текстового файла (ERROR, PROCESS, SUCCESS).

:Пример:

ERROR

Содержимое файла в случае ошибки.

Обновление системы cl-update

- [Обновление системы cl-update](#)
- [Описание функциональных возможностей](#)
- [Описание опций утилиты](#)
- [Ключи для управления синхронизацией и кэшами](#)
- [Ключи применения шаблонов, ревизий](#)
- [Другие ключи](#)

cl-update - утилита, выполняющая обновление системы со всеми необходимыми сопутствующими действиями.

Описание функциональных возможностей

Порядок обновления в общем случае следующий:

1. Синхронизация репозиториев дистрибутива
2. Если репозитории были обновлены, то выполняются действия *egencache* и *eix-update*
3. Обновление ревизии и обновление мира
4. Обновление системы
5. Обновление Python/Perl с пересборкой поврежденных пакетов при необходимости
6. Удаление ненужных пакетов
7. Пересборка модулей ядра при необходимости
8. Пересборка прочих поврежденных пакетов при необходимости
9. Пересборка пакетов для Xorg-сервера, если в этом есть необходимость
10. Выполнение *dispatch-conf*

Описание опций утилиты

При запуске cl-update без параметров обновление будет происходить по общей схеме, но можно изменить поведение утилиты, используя ключи.

Ключи для управления синхронизацией и кэшами

-r REPOSITORIES, --repositories REPOSITORIES

Задает синхронизируемые репозитории дистрибутива (все по умолчанию) ('list' для отображения возможных значений). При указании опции позволяет синхронизировать лишь выбранный репозиторий.
--branch BRANCHES

Устанавливает ветки для репозиториев (REPOSITORY:BRANCH). Позволяет переключать между master/develop/update ветками репозиториев.

--force-egencache

Принудительно обновляет кэш оверлеев.

--skip-eengcache

Пропускает обновление кэша оверлеев.

--auto-eengcache

Обновляет кэш оверлеев в том случае, если тот устарел (действие по умолчанию).

--force-eix-update

Принудительно обновляет кэш eix.

--skip-eix-update

Пропускает обновление кэша eix.

--auto-eix-update

Обновляет кэш eix, если он устарел (поведение по умолчанию).

-o [ON/OFF], --update-other [ON/OFF]

Выполняет обновление сторонних оверлеев (не являющихся репозиториями дистрибутива). По умолчанию выключено. Для того, чтобы изменить действие по умолчанию, надо изменить значение переменной `cl_update_other_set` секции `update`; для этого можно выполнить следующую команду:

`cl-core-variables --set update.cl_update_other_set=on`

-s [ON/OFF], --sync-only [ON/OFF]

Не обновлять пакеты. Выполняет синхронизацию и применяет ревизии.

Ключи применения шаблонов, ревизий

--rebuild-world [ON/OFF]

Переформирует `world`-файл согласно шаблонам дистрибутива на `world`-файл по умолчанию для данного профиля.

--update-rev [ON/OFF]

Обновляет ревизию дистрибутива до текущей в шаблонах.

-T TEMPLATES, --templates TEMPLATES

Выбор местонахождения шаблонов `calculate, distros, local, remote, clt` ('list' для отображения возможных значений)

--usenew-conf

Использовать новые конфигурационные файлы.

--skip-conf

Пропустить обновление конфигурационных файлов.

--dispatch-conf

Обновить конфигурационные файлы вручную (по умолчанию).

Другие ключи

-p [ON/OFF], --pretend [ON/OFF]

Вместо действительного обновления пакетов, только отобразить, что будет установлено.

--wait-another-update [ON/OFF]

Ждать завершения выполнения другого запущенного cl-update. По умолчанию включено.

--schedule [ON/OFF]

Учитывать график автопроверки обновлений. Если временной интервал с последней проверки ещё не прошел, то обновление выполняться не будет.

-v [ON/OFF], --verbose [ON/OFF]

Включает подробный вывод.

-f, --force

Не задавать вопросы во время процесса.

Смена профиля системы cl-update-profile

Описание утилиты

cl-update-profile - утилита предназначена для смены репозитория и профиля системы.

В общем виде утилита используется следующим образом:

cl-update-profile [--url CL_UPDATE_PROFILE_REPO] [-s [ON/OFF]] [-f] [PROFILE]

Описание опций

Установить репозиторий профиля

--url CL_UPDATE_PROFILE_REPO

Репозиторий указывается следующим образом (используется либо вместе с именем профиля, либо с опцией list):

cl-update-profile --url=git://git.calculate.ru/calculate/distros.git list

Команда выведет список профилей данного репозитория:

Профиль системы:

[CLD] distros:CLD/amd64
[CLDX] distros:CLDX/amd64
[CLS] distros:CLS/amd64
[CMC] distros:CMC/amd64
[CDS] distros:CDS/amd64
[CSS] distros:CSS/amd64

Для установки профиля аналогично предыдущей команде, но используя имя профиля:

cl-update-profile --url=git://git.calculate.ru/calculate/distros.git
distros:CLDX/amd64

Вывод команды:

Репозиторий

* Название репозитория: distros

Профиль

```
* Профиль системы: distros:CLDX/amd64
* Название дистрибутива: Calculate Linux Desktop 15 Xfce
* Используемые репозитории:
+-----+-----+
|Название |URL
+-----+-----+
|distros |git://git.calculate.ru/calculate/distros.git|
|calculate|git://git.calculate.ru/calculate/overlay.git|
|portage |git://git.calculate.ru/calculate/portage.git|
+-----+-----+
* Список пакетов системы: Обновить
* Пропустить настройку системы: нет
```

Запустить процесс? (Yes/No): yes

Синхронизация репозиториев

```
* Синхронизация Distros репозитория ...
[ ok ]
* Синхронизация Calculate репозитория ...
[ ok ]
* Синхронизация Portage репозитория ...
[ ok ]
* Синхронизация завершена
Настройка профиля
* Переключение на CLDX профиль ...
[ ok ]
* Обновление профиля завершено успешно
```

Для сервисов github и bitbucket поддерживается укороченный формат. Пример с использование стороннего пользовательского репозитория:

```
cl-update-profile --url=github:lautre76 list
```

Синхронизировать или нет репозитории:

```
-s [ON/OFF], --sync [ON/OFF]
```

Не задавать вопросы во время процесса:

```
-f, --force
```

Подробно о создании своего профиля и использовании утилиты можно прочитать в [блоге](#).

Сборка системы

- [Сборка системы](#)
- [Введение](#)
- [Необходимые требования](#)
- [Использование](#)
- [Подготовка системы к сборке](#)
- [Смена профиля системы](#)
- [Обновление системы](#)
- [Создание загрузочного образа](#)
- [Прерывание сборки](#)
- [Восстановление сборки](#)
- [Создание мультизагрузочной Live USB Flash](#)

Введение

Все операции по сборке образов дистрибутива Calculate Linux выполняются при помощи утилит Calculate. По завершению сборки создаётся загрузочный образ Live USB, который можно использовать как для работы, так и для установки системы на жёсткий диск компьютера.

Начиная с версии утилит 3.4, модуль сборки системы был полностью переписан. Выполняемые файлы утилит были переименованы и теперь начинаются с префикса 'cl-builder-', после которого следует выполняемое действие (например, 'cl-builder-update').

Необходимые требования

Для выполнения сборки вам понадобится выход в интернет, свободное место на диске, а также любой доступный "ISO-образ системы Calculate Linux": версии 15 или выше, например, [Calculate Linux Scratch](#) или Calculate Scratch Server. Вместо ISO-образа можно указать CD-привод или USB Flash с установленным Calculate Linux. Для больших возможностей в формировании собственных сборок неплохо иметь [свой Git-репозиторий](#) с Calculate-совместимым профилем системы. Для размещения репозитория подойдёт такой хостинг как [GitHub](#), [Bitbucket](#) или любой другой, например, собственный, поднятый при помощи [Gitolite](#).

Для разворачивания и обновления образа без использования слоёв вам может потребоваться около 5 Гб свободного места на жёстком диске и 15 Гб при разворачивании образа без использовании слоёв (см. Подготовка системы к сборке). Для создания образа дистрибутива потребуется около 2 Гб. В случае создания бинарных пакетов дополнительно может потребоваться до 5 Гб свободного места.

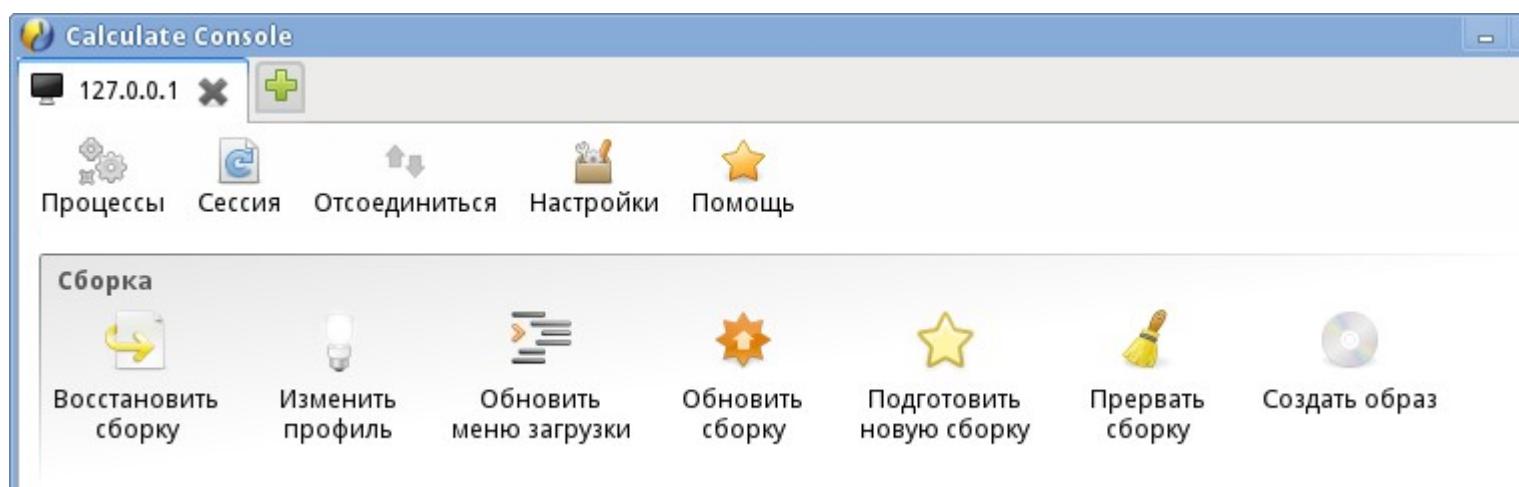
Сборка дистрибутива может производиться как на отдельном разделе жёсткого диска, так и в выделенном каталоге файловой системы. Готовый образ сохраняется в директории `/var/calculate/linux`, бинарные пакеты сохраняются в директории `/var/calculate/remote/builder`. Позаботьтесь о наличии свободного места по этим путям.

Использование

Пакет `sys-apps/calculate-utils` 3.4, входящий во все дистрибутивы Calculate Linux 15, включает в себя необходимые компоненты для сборки системы. **Основные возможности:**

- поддержка графического интерфейса и работы из командной строки;
- поддержка работы с многослойной файловой системой OverlayFS;
- поддержка работы в системе, загруженной с Live USB;
- поддержка параллельной сборки нескольких дистрибутивов;
- создание мультизагрузочных USB Flash;
- профили сборки: Calculate-совместимые;
- поддерживаемые архитектуры: i686 и x86_64;
- поддержка сборки 32-битных дистрибутивов на 64-битной системе.

Операции по сборке системы выполняются в разделе "Сборка" графической консоли утилит Calculate:



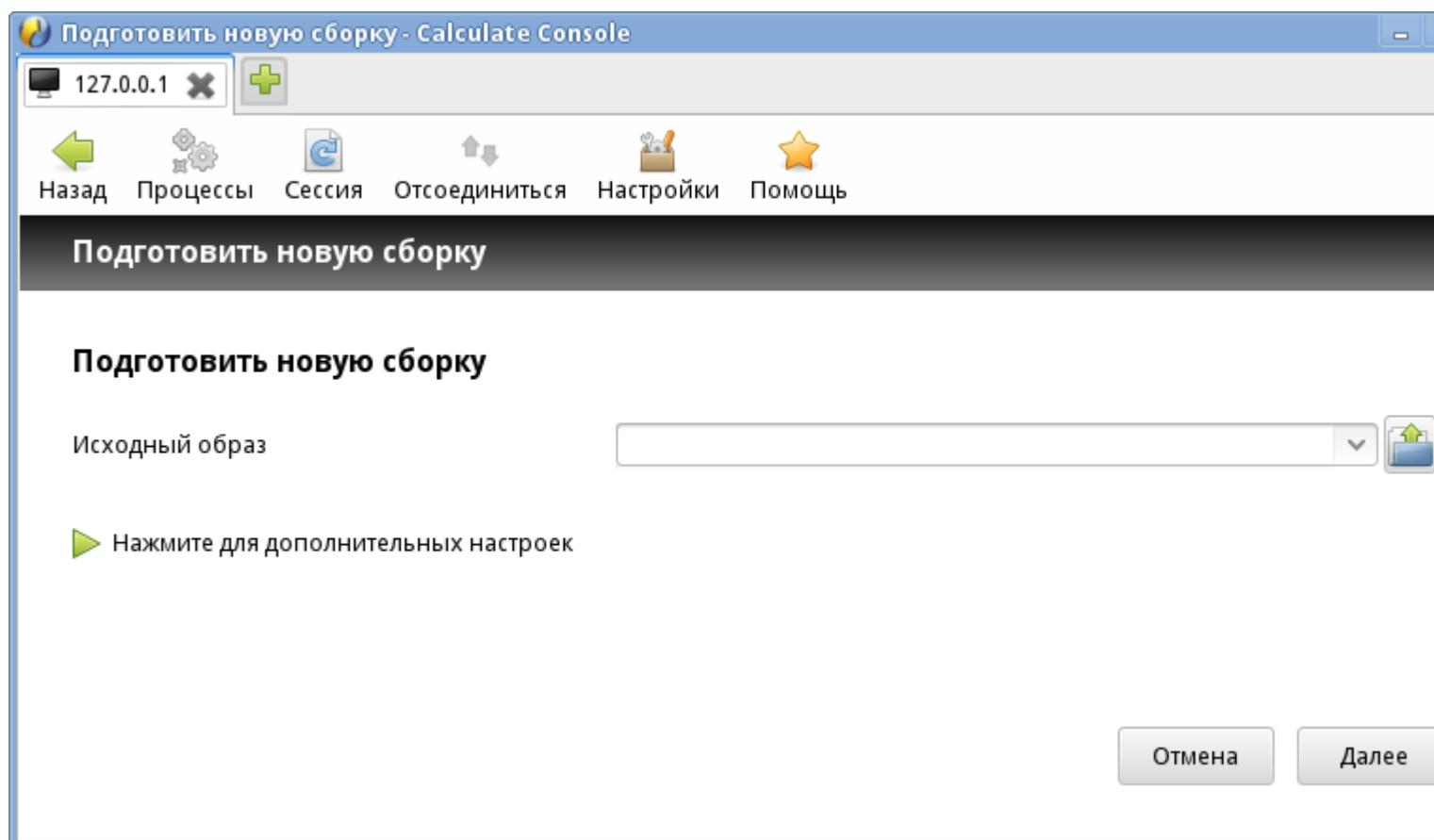
Для работы из командной строки вы можете использовать следующие утилиты:

- **cl-builder-prepare** (Подготовить новую сборку) - используется для подготовки данных для сборки;
- **cl-builder-profile** (Изменить профиль) - используется для смены профиля собираемой системы;
- **cl-builder-update** (Обновить сборку) - используется для обновления пакетов собираемой системы;
- **cl-builder-break** (Прервать сборку) - используется для прекращения сборки;
- **cl-builder-image** (Создать образ) - используется для создания загрузочного ISO-образа;
- **cl-builder-restore** (Восстановить сборку) - используется для восстановления данных сборки после перезагрузки машины;
- **cl-builder-menu** (Обновить меню загрузки) - используется для обновления мультизагрузочного меню Live USB.

Вызов утилит из консоли можно сочетать с работой графических утилит, т.к. они остаются полностью совместимыми.

Подготовка системы к сборке

Для подготовки системы к сборке кликните по иконке "Подготовить новую сборку" в разделе "Сборка" графической консоли утилит Calculate:



В открывшейся странице выберите исходный образ системы. В случае загрузки с Live USB по умолчанию вам будет предложен загруженный образ. По необходимости используйте дополнительные настройки.

Для работы в терминале выполните:

```
cl-builder-prepare
```

Основные опции:

- **--source SOURCE** - исходный образ системы.

В качестве исходного образа может выступать ISO образ, CD-привод или Live USB с дистрибутивом Calculate Linux. Дополнительные опции:

- **-d DEST, --disk DEST** - раздел или директория для сборки. Убедитесь, что в разделе и в каталоге у вас не хранится информации, так как во время подготовки сборки она будет удалена.
- **--layers [ON/OFF]** - использовать многослойную файловую систему (OverlayFS) для подготовки сборки, при использовании этой опции образ не будет распакован в файловую систему, в файловой системе будут храниться только изменения от базового образа. Не используется при сборке в отдельном разделе.
- **--id ID** - идентификатор сборки. При одновременной сборке нескольких систем служит для обозначения сборки. По умолчанию обозначается как **имя_репозитория:имя_профиля**.

Исходя из того, укажете ли вы раздел или директорию сборки, будете ли вы использовать многослойную файловую систему, у вас может быть три варианта подготовки образа:

1. Сборка в выделенной директории с использованием многослойной файловой системы.
2. Сборка в выделенной директории без использования многослойной файловой системы.
3. Сборка в выделенном разделе диска.

Сборка в выделенном разделе с использованием многослойной файловой системы не поддерживается.

Для экономии времени и места, по умолчанию, при подготовке образа используется многослойная файловая система.

Во всех типах сборки доступ к корню собираемой системы можно получить в директории **/run/calculate/mount/каталог_сборки**, где каталог сборки - это имя сборки с подменой символов ":" и "/" на "_" (пример: "distros_CLSK_amd64").

Во время подготовки системы к сборке выполняются следующие действия:

1. Подключение базового образа в директории **/var/calculate/mount/iso** и **/var/calculate/mount/squash**
2. Распаковка базового образа
3. Настройка шаблонами *builder/prepare* (событие *ac_builder_prepare*), *builder/setup* (событие *ac_builder_setup*)
4. Подключение точек монтирования **/dev**, **/dev/shm**, **/dev/pts**, **/proc**, **/sys**, **/var/calculate/remote**
5. Отключение базового образа системы

Особенности в подготовке системы к сборке при использовании выделенного раздела:

- Для развёртывания образа потребуется выделенный раздел, данные в котором будут удалены при форматировании. В качестве файловой системы будет использована текущая файловая система, если она поддерживается утилитами.

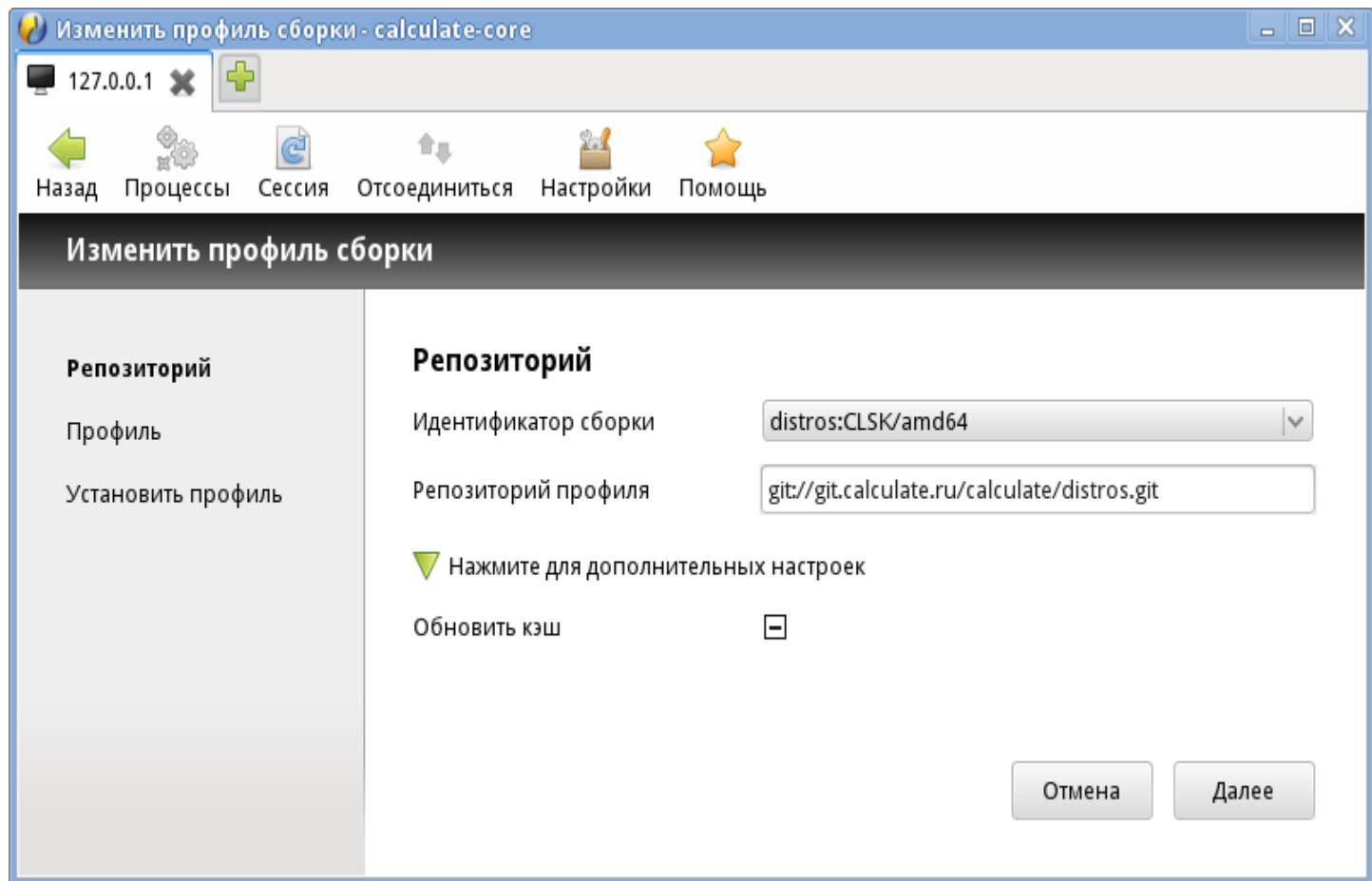
Особенности в подготовке системы к сборке при использовании слоёв:

- Вместо распаковки исходный образ монтируется и остаётся примонтированным на всём протяжении сборки системы.
- Дельта выполняемых изменений хранится в директории **/var/calculate/builder/каталог_сборки**.

Смена профиля системы

Смена профиля собираемой системы вам может понадобиться, главным образом, для выбора своего профиля, в котором вы можете описать особенности системы, такие как: состав пакетов, используемые флаги сборки, опции компилятора, настройки переменных утилит Calculate, шаблоны настройки системы, ebuild-ы пакетов и т.д.

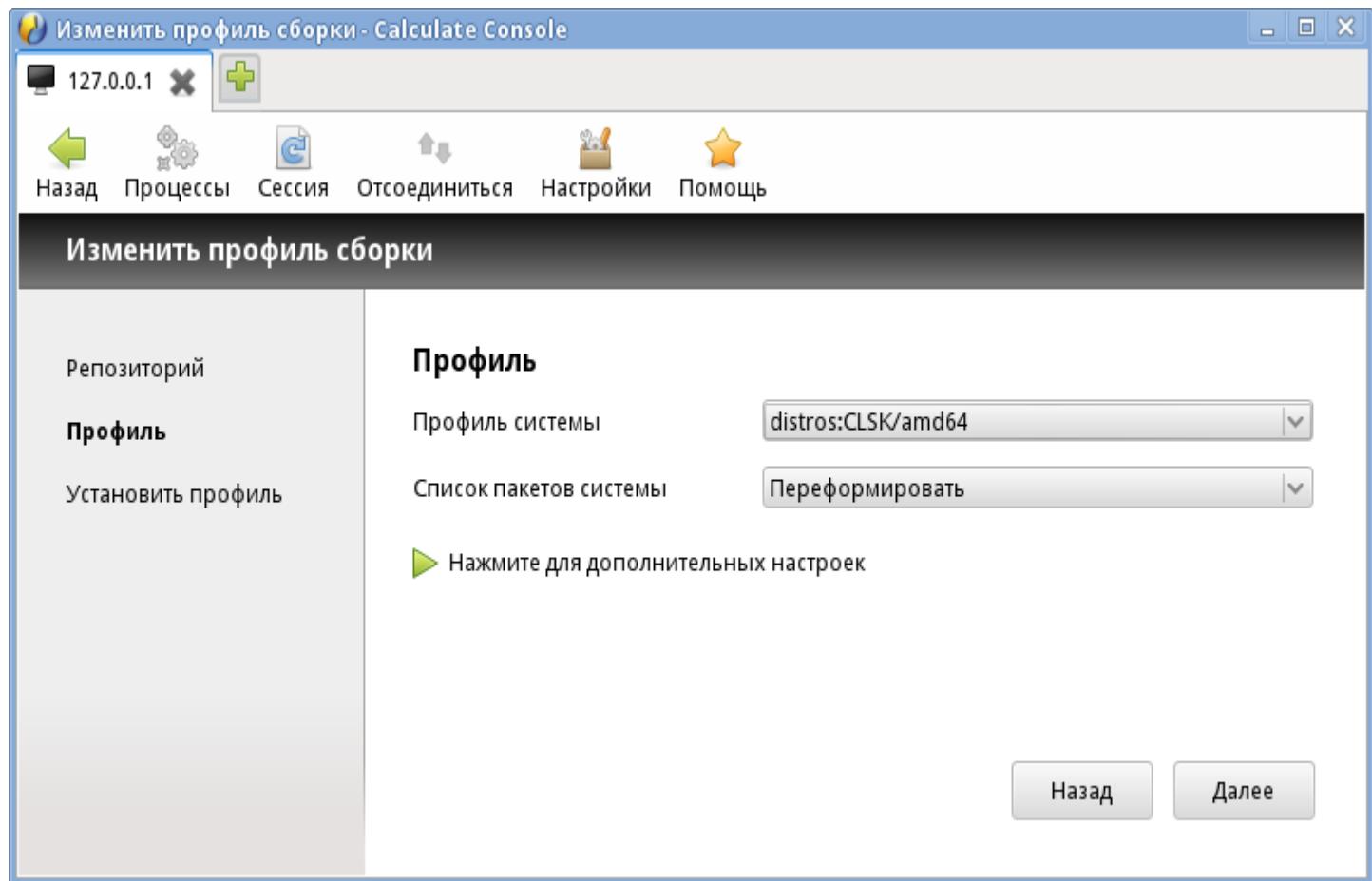
Для переключения профиля, воспользуйтесь иконкой "Изменить профиль" в разделе "Сборка" графической консоли утилит, после чего откроется следующая страница:



По умолчанию отображаются профили репозитория distros. При использовании своего профиля, хранящегося, например, на Github, вы можете указать "github:lautre76", где "github" - обозначение сервиса хостинга, "lautre76" - имя вашей учётной записи. При указании сокращённого имени поиск профилей будет выполняться в репозитории "overlay.git".

В дополнительных настройках вы можете указать "Обновить кэш" для того, чтобы при наличии кэша репозитория данные были обновлены. Это может понадобиться, если вы захотите добавить новый профиль, отсутствовавший при предыдущем вызове команды.

После нажатия кнопки "Далее" откроется окно выбора профиля:



В списке доступных профилей отображаются все профили выбранного репозитория.

Состав пакетов (файл `/var/lib/portage/world`) формируется шаблонами утилит Calculate. Вы можете выбрать один из режимов обновления файла: переформировать, объединить или обновить. По умолчанию используется полная замена списка пакетов дистрибутива.

При работе в терминале для смены профиля вы можете использовать следующую команду:
`cl-builder-profile`

Основные параметры:

- **--id ID** - идентификатор сборки. Указывать данный параметр необходимо в случае одновременных сборок. Просмотреть список идентификатором можно при помощи значения `list`.
- **--url URL** - репозиторий профиля. По умолчанию использует репозиторий, в котором находится текущий профиль системы.
- **--rebuild-world, --merge-world, --update-world** - действие с файлами `world` при переключении профиля: список замещается списком из профиля, текущий список объединяется с новым или выполняется только обновление списка пакетов.

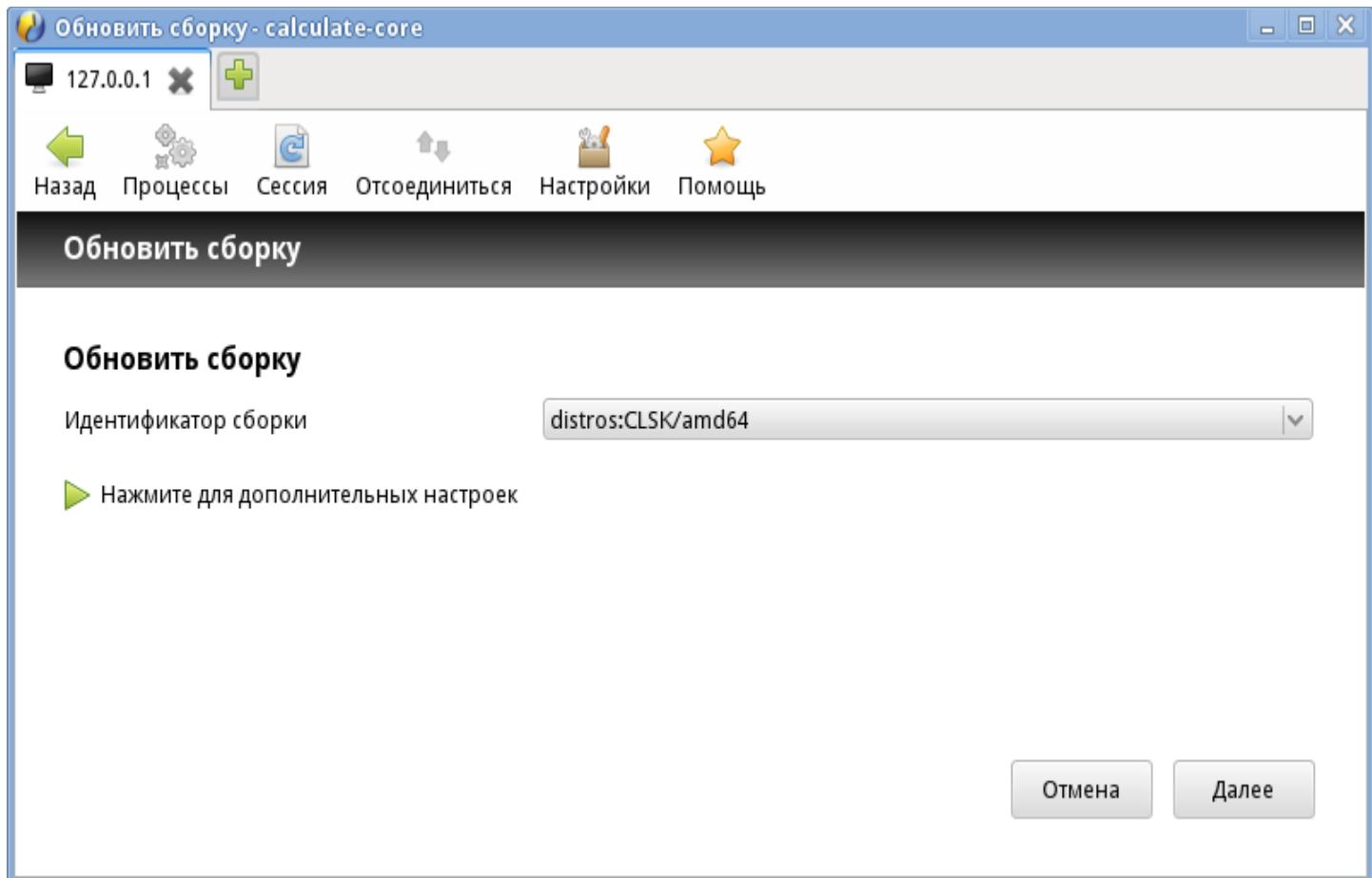
Дополнительные параметры:

- **-u [ON/OFF], --update-cache [ON/OFF]** - обновить скачанный ранее репозиторий
- **--skip-setup-system [ON/OFF]** - пропустить перенастройку системы после переключения профиля. По умолчанию после переключения профиля в системе выполняется действие аналогичное `cl-setup-system`, за исключением того, что в собираемой системе не будут выполнены шаблоны события `3_ac_install_live`.

Переключение профиля в собираемой системе по сути аналогично переключению профиля в текущей системе при помощи команды `cl-update-profile`.

Обновление системы

Для обновления пакетов собираемой системы кликните по иконке "Обновить сборку" в разделе "Сборка" графической консоли утилит Calculate. После чего вам откроется следующая страница:



Для обновления собираемой системы из терминала используйте следующую команду:
cl-builder-update

Параметры:

- **--id ID** - идентификатор сборки. Указывать данный параметр необходимо в случае одновременных сборок.
- **-s [ON/OFF], --sync-only [ON/OFF]** - выполнить только синхронизацию репозиториев и обновление настроек
- **-o [ON/OFF], --update-other [ON/OFF]** - обновление вспомогательных репозиториев
- **-p [ON/OFF], --pretend [ON/OFF]** - вместо действительного обновления пакетов отобразить только, что будет установлено
- **-r REPOSITORIES, --repositories REPOSITORIES** - синхронизируемые репозитории. По умолчанию синхронизируются все репозитории дистрибутива, причём если репозиторий бинарный, то синхронизация будет до необходимых ревизий, в противном случае до ветки master.
- **-e [ON/OFF], --emergelist [ON/OFF]** - отобразить список пакетов в формате emerge
- **--(rebuild|merge|update)-world** действие с файлов world: список замещается базовым набором, текущий список объединяется с базовым или выполняются только обновление списка пакетов
- **--(force|skip|auto)-egencache** - обновить кэш репозиториев: принудительно, не обновлять, при необходимости.
- **--(force|skip|auto)-eix-update** - обновить кэш eix: принудительно, не обновлять, при необходимости.
- **--rebuild-changed-packages [ON/OFF]** - пересобрать пакеты, ebuild файлы которых изменились (изменились либо сами ebuild файлы, либо eclass файлы, используемые для сборки пакетов).

- **-R [ON/OFF], --skip-revdep-rebuild [ON/OFF]** - пропустить выполнение команды revdep-rebuild (опция включена по умолчанию)
- **--scan [ON/OFF]** - выполнить поиск наиболее актуального сервера бинарных обновлений
- **--clean-pkg [ON/OFF]** - удалять устаревшие архивы программ (очистка packages и distfiles от версий пакетов, которые отсутствуют в дереве portage)
- **--branch REFS** - переключить репозитории на указанные ветки или ревизии. Начиная с версии 3.4 недостаточно переключить на нужную ветку один раз. При последующем запуске без параметра **--branch** утилиты попытаются привести репозитории к ревизиям, указанным на выбранном сервере бинарных обновлений.

Функционал cl-builder-update повторяет функционал cl-update за исключением того, что обновление выполняется внутри подготавливаемой сборки. **Список выполняемых действий:**

Действия, обновляющие репозитории и настройки:

1. Настройка шаблонами *builder/prepare* (событие *ac_builder_prepare*), *builder/setup* (событие *ac_builder_setup*)
2. Синхронизация репозиториев (можно пропустить, указав "-r none")
3. Синхронизация прочих оверлеев (можно пропустить, указав "-o off")
4. Обновление кэша метаданных в репозиториях (можно пропустить, указав "--skip-eigen-cache")
5. Обновление кэшей связанных с репозиториями (можно пропустить указав "--skip-eix-update")
6. Удаление устаревших файлов из distfiles, packages (выполняется при "--clean-pkg")
7. Исправление настроек в собираемой системе (шаблоны события *ac_update_sync*)
8. Выполнение dispatch-conf Действия, обновляющие пакеты в системе, выполняются, если не указана опция -p:
9. Выполнение emerge -uDN --changed-deps --with-bdeps=y @world
10. Пересборка изменившихся пакетов
11. Обновление Python пакетов
12. Обновление Perl пакетов
13. Выполнение emerge --depclean
14. Пересборка модулей ядра
15. Пересборка модулей xorg-server
16. Пересборка @preserved-libs
17. Выполнение revdev-rebuild
18. Выполнение dispatch-conf
19. Исправление бинарных пакетов
20. Отображение списка новостей
21. Проверка на устаревшие пакеты

Отдельной команды для проверки зависимостей нет, их можно проверить без обновления репозиториев при помощи cl-update -o off -r none -p.

Во время сборки будут загружены все необходимые пакеты с исходными текстами программ в директорию /var/calculate/remote/distfiles.

Помните, что вы всегда можете получить доступ к системе при помощи chroot, выполнив:

```
chroot /run/calculate/mount/каталог_сборки
```

Пример:

```
chroot /run/calculate/mount/distros_CLDX_amd64
```

Обратите внимание, что для доступа к собираемой 32-битной системе из-под 64-битной команду chroot следует выполнять, используя утилиту linux32. Пример:

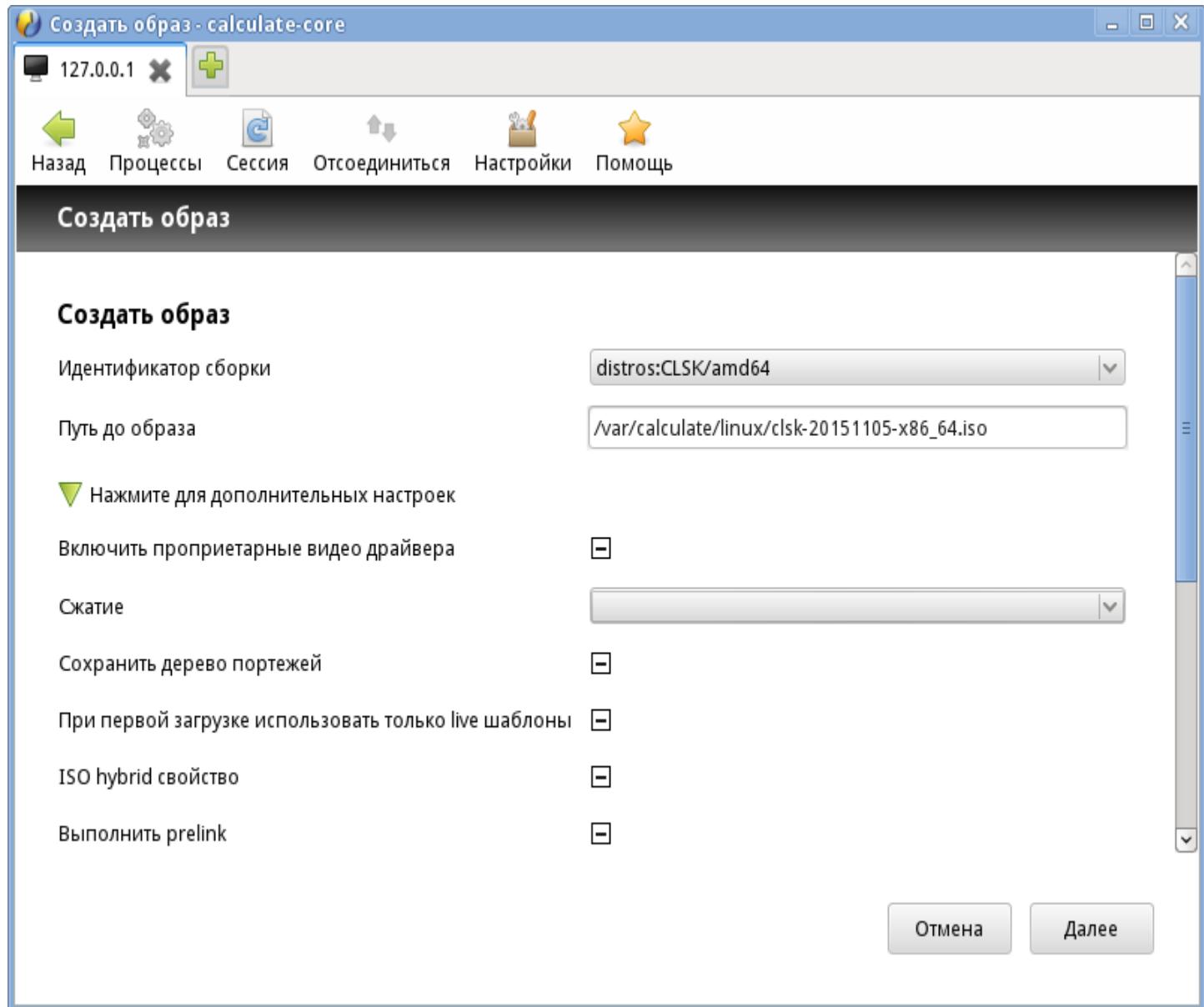
```
linux32 chroot /run/calculate/mount/distros_CLDX_i686
```

После завершения действий в chroot из оболочки необходимо выйти, в противном случае выполнение действий по обновлению системы, смене профиля или создания образа завершится сообщением об ошибке. Пример:

```
* /bin/bash уже запущена внутри distros:CLSK/amd64
```

Создание загрузочного образа

Для создания нового образа дистрибутива воспользуйтесь иконкой "Создать образ" в разделе "Сборка" графической консоли утилит Calculate. Страница создания образа будет выглядеть, как на рисунке ниже:



Большинство настроек будет по умолчанию скрыто от глаз. При желании вы можете указать, включить ли дистрибутивы проприетарных видеодрайверов в образ Live USB для того, чтобы иметь возможность оценить их работу, загрузившись с USB Flash. Вы можете выбрать метод сжатия образа для того, чтобы найти оптимальный вариант между размером файла и скоростью сжатия. Опция ISO hybrid позволит создавать образ диска, который можно писать на устройство при помощи прямого копирования (например, dd). Выполнение Prelink позволит слинковать библиотеки перед запаковкой для получения некоторого ускорения запуска программ.

После выполнения в директории /var/calculate/linux/ будет создан образ дистрибутива (с расширением .iso), файл с контрольными суммами (с расширением DIGESTS) и файл с составом программ (с расширением list).

В терминале запаковать образ можно, выполнив:

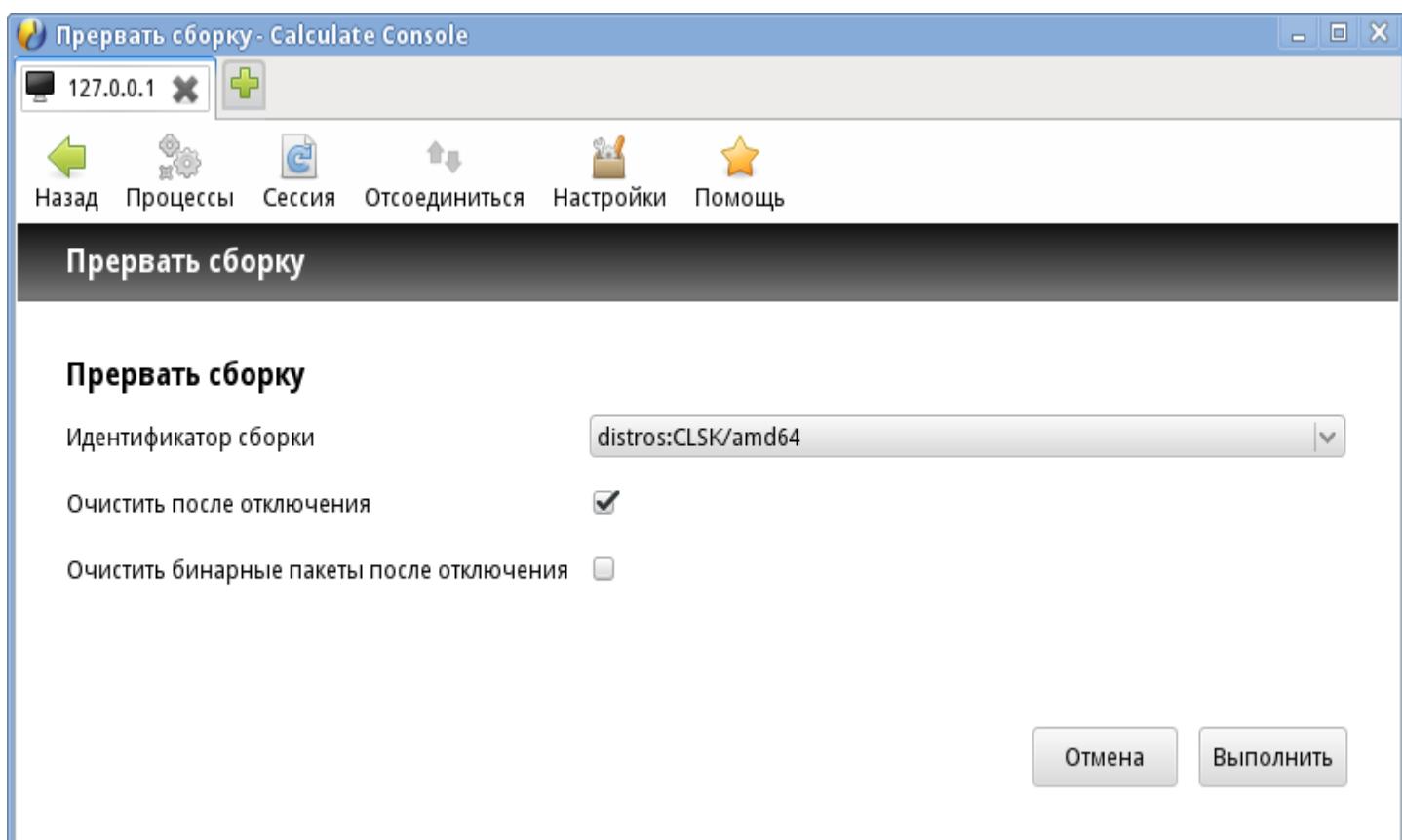
Основные параметры:

- **--id ID** - идентификатор сборки; необходим, только если производится несколько сборок одновременно.
- **--iso IMAGE** - путь и имя создаваемого ISO образа. Если не указывать, то образ будет помещен в каталог по умолчанию, и имя будет в стандартном формате, содержащее короткое имя дистрибутива, архитектуру и build. **Дополнительные параметры:**
- **-V, --video [ON/OFF]** - включить установочные файлы proprietарных драйверов в дистрибутив. При выключенном опции удаляется из дистрибутива установочные файлы proprietарных драйверов.
- **-c, --compress COMPRESS** - формат сжатия при подготовке squashFS
- **--keep-tree [ON/OFF]** - сохранить дерево портежей в образе. По умолчанию из дерева портежей (и оверлеев) удаляются ebuild файлы и metadata, остаются только eclass -файлы, профили и шаблоны (за исключением шаблонов в каталоге deprecated).
- **--live [ON/OFF]** - выполнить модификацию init.d/calculate, так, чтобы при первой загрузке выполнялись только шаблоны ac_install_live. Опция включена по умолчанию, так как при сборке дистрибутива шаблоны ac_install_merge выполняются на этапе сборки пакета.
- **--isohybrid [ON/OFF]** - модифицировать по завершению полученный образ утилитой isohybrid, чтобы его можно было записать USB Flash командной dd
- **--prelink [ON/OFF]** - выполнить prelink перед созданием ISO образа

Полученный ISO образ в дальнейшем можно использовать в том числе и для новых сборок.

Прерывание сборки

Чтобы прервать выполнение сборки, воспользуйтесь иконкой "Прервать сборку" в разделе "Сборка" графической консоли утилит. После этого откроется следующая страница:



После выполнения операции все временные файлы будут удалены.

В терминале прервать сборку можно, выполнив:

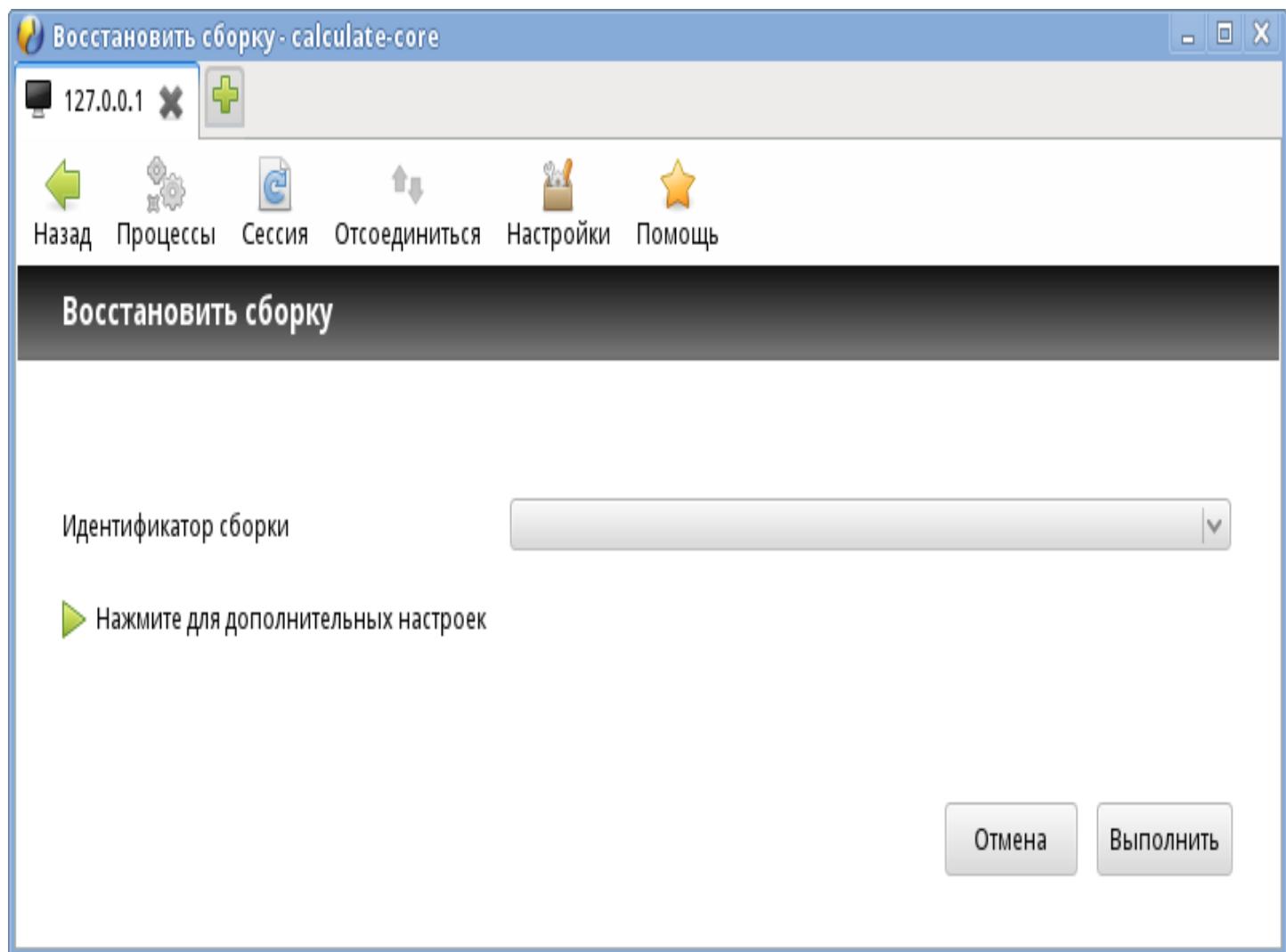
`cl-builder-break`

Параметры:

- `--id ID` - идентификатор сборки; необходим, только если производится несколько сборок одновременно.
- `--clear [ON/OFF]` - очистить данные после отключения сборки (актуально при сборке в отдельном каталоге). Опция включена по умолчанию.
- `--clear-pkg [ON/OFF]` - удалить бинарные пакеты

Восстановление сборки

Если во время работы с дистрибутивом компьютер был перезагружен, вы можете выполнить восстановление прерванной сборки. Для этого кликните по иконке "Восстановить сборку" в разделе "Сборка" графической консоли утилиты Calculate. После чего откроется следующее окно:



То же действие можно выполнить в терминале, набрав:

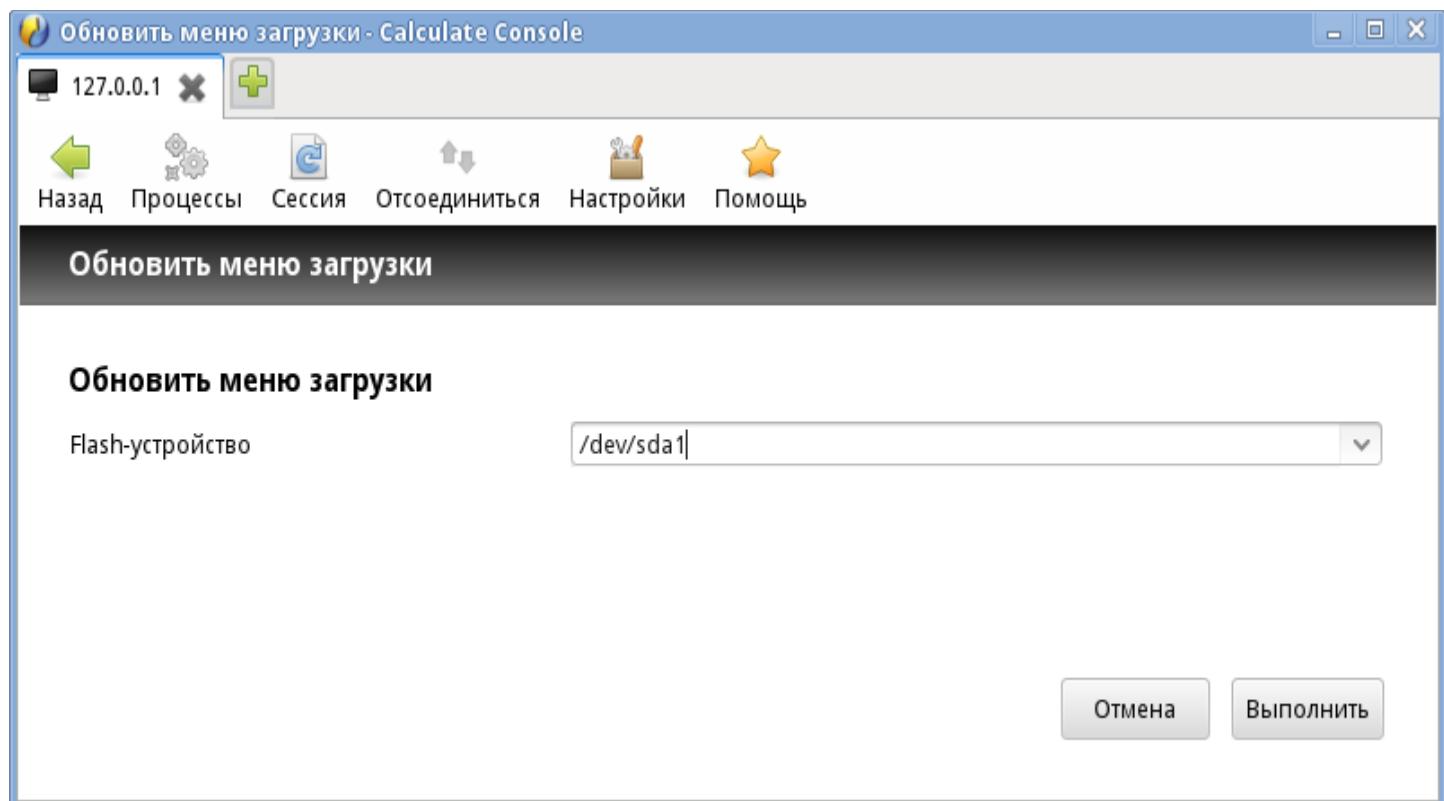
`cl-builder-restore`

В качестве параметра вы можете указать идентификатор сборки.

Создание мультизагрузочной Live USB Flash

Вы можете полноценно работать с Live USB, создавая модификации текущего, либо любого другого ISO образа Calculate Linux. Для возможности сохранять на USB Flash данные система должна быть установлена на флешку при помощи утилиты Calculate.

Во время работы с Live USB при создании образа загрузочное меню переформировывается таким образом, чтобы можно было выбрать загрузку с любого из созданных ISO-образов. Вы также можете просто скопировать в директорию /linux USB Flash другие ISO-образы Calculate Linux, а затем обновить меню загрузчика. Для этого кликните по иконке "Обновить меню загрузки" в разделе "Сборка" графической консоли утилит Calculate. После этого откроется следующая страница:



Вы можете сформировать меню флешки, загрузившись с неё, либо вставив в USB разъём и из текущей системы при помощи утилиты выполнить обновление.

При работе в терминале для обновления меню выполните:

```
c1-builder-menu -d устройство
```

Вместо устройства можно указать путь, в который подключена USB Flash.

Calculate API

- [Calculate API](#)
- [Введение](#)
- [Состав Calculate API](#)
- [Основные модули](#)
- [Метод, запускающий процесс на выполнение](#)
- [Метод, предоставляющий дополнительную информацию](#)
- [Метод процесса](#)
- [Категория Установка](#)
- [Категория Настройка](#)
- [Категория Утилиты](#)
- [Вспомогательные модули](#)
- [Методы работы с сертификатами](#)
- [Методы работы с сессиями](#)
- [Методы работы с процессами](#)
- [Методы работы с кэшем](#)
- [Прочие вспомогательные методы](#)

Введение

Calculate API (Интерфейс прикладного программирования утилит Calculate) - набор готовых классов, функций, структур и констант утилит Calculate, предоставляемых для использования во внешних программных продуктах. Calculate API предоставляет инструменты взаимодействия различных клиентов (таких как [cl-console](#) и [cl-console-gui](#)) с [сервером утилит cl-core](#). Подробнее см. [API в Википедии](#).

Доступ к API осуществляется через протокол HTTPS, использующий криптографический протокол SSL, который зашифровывает все передаваемые данные и этим защищает их от несанкционированного доступа. Установка SSL-соединения обязательна для вызова API методов.

В утилитах Calculate взаимодействие осуществляется по протоколу [SOAP](#). Обмен данными по этому протоколу ведется посредством XML-сообщений.

Состав Calculate API

На стороне сервера утилит API реализовано с использованием библиотеки [dev-python/soaplib](#).

Все модули и функции Calculate API можно разделить на две большие группы:

- **Основные** - служат для вызова методов, непосредственно выполняющих выбранные пользователем действия (установка системы, настройка сети и др.);
- **Вспомогательные** - служат для обеспечения взаимодействия клиента и сервера утилит (получение списка доступных методов, получение сессии, проверка прав сертификата и др.).

API реализовано с помощью [rpc](#) декораторов библиотеки soaplib, осуществляющих взаимодействие и проверку параметров (их число и типы данных).

Основные модули

Основные модули и методы Calculate API осуществляют основные действия сервера утилит.

Простейшие из них представляют собой отдельный класс с именем CoreWsdl (для импортирования в сервер утилит). В этом классе находятся как минимум три метода (два из которых [rpc](#) методы), связанных между собой:

- метод, запускающий процесс на выполнение;
- метод, предоставляющий дополнительную информацию клиенту;
- метод, выполняющий выбранное пользователем действие (представляет собой отдельный процесс).

Метод, запускающий процесс на выполнение

Рассмотрим на примере метода настройки пакетов в системе core_setup.

Имя метода, запускающего процесс на выполнение, является именем, по которому клиент вызывает данный метод (запускает процесс), в нашем случае это `_core_setup_`. Основные функции методов, запускающих процесс:

- проверка поступающих от клиента параметров и сообщение об ошибках при необходимости;
- создание глобального словаря параметров процесса для хранения всей информации о процессе;
- запуск процесса на выполнение.

Каждый такой метод обрамляется двумя обязательными декораторами (подробнее о декораторах на [ru.wikipedia.org](#)): [rpc](#) и [core_method](#).

Декоратор rpc

Декоратор [rpc](#) служит для указания типа входных и возвращаемых данных. В декораторе метода `core_setup` два входных параметра - целочисленный (номер сессии) и типа `CoreSetupInfo` (объект, хранящий необходимые значения для запуска и выполнения процесса), и один возвращаемый параметр - массив элементов, каждый из которых имеет тип `ReturnedMessage`:

```
@rpc(Integer, CoreSetupInfo, _returns = Array(ReturnedMessage))
```

Возвращаемый тип ReturnedMessage

Для сообщения клиенту об успешном запуске процесса или об ошибках сервер утилит возвращает массив типовых сообщений типа ReturnedMessage. Каждое сообщение состоит из следующих полей:

- **type** - тип сообщения. В случае успешного запуска процесса type рано строке "pid", в случае ошибки "error", при необходимости сообщить предупреждение - "warning"
- **message** - передаваемое сообщение. В случае успешного запуска процесса значение message равно идентификатору процесса (pid), в остальных случаях message хранит строку с сообщением об ошибке или предупреждении;
- **field** - поле с ошибкой или предупреждением. Значение равно имени параметра, в котором допущена ошибка или для которого необходимо сообщить предупреждение;
- **expert** - устанавливается, если поле находится в дополнительных настройках.

Декоратор core_method

Декоратор **core_method** используется для указания дополнительной информации о методе для клиента и имеет шесть параметров:

- **category** - указывает в какой категории будет находиться метод (для графических клиентов);
- **title** - указывает отображаемое название метода, как правило является строкой для перевода;
- **image** - указывает иконки для отображения (используется для графических клиентов. Иконки могут перечисляться через запятую и клиент использует первую найденную);
- **gui** - определяет, используется ли данный метод для графических клиентов;
- **command** - команда для создания символьской ссылки на сервер утилит cl-core. Также указывает, используется ли данный метод для консольного клиента;
- **rights** - указывает список прав, необходимых для возможности запуска данного метода. Для успешного запуска метода необходимы все указанные права.

Для метода core_setup:

```
@core_method(category=__('Configuration'), title=__('Configure package'),
              image='applications-other', gui=True, command='cl-core-
setup',
              rights=['configure'])
```

Декораторы указываются непосредственно перед объявлением метода и выполняются один раз. В итоге метод core_setup имеет вид:

```
...
class CoreWsdl:
    @rpc(Integer, CoreSetupInfo, _returns = Array(ReturnedMessage))
    @core_method(category=__('Configuration'), title=__('Configure
package'),
                 image='applications-other', gui=True, command='cl-core-
setup',
                 rights=['configure'])
    def core_setup(self, sid, info):
        ...
...
```

Метод, предоставляющий дополнительную информацию

Рассмотрим на примере метода настройки пакетов в системе core_setup.

Для каждого метода, запускающего процесс на выполнение, должен быть метод, который сообщает клиенту информацию об используемых параметрах. Имя такого метода составляется из имени основного метода и окончания *_view*, например для метода core_setup имя метода, предоставляющего дополнительную информацию - core_setup_view.

Методы, предоставляющие дополнительную информацию клиентам, имеет один декоратор `rpc`, служащий для указания типа входных и возвращаемых данных. Для метода `core_setup_view` декоратор `rpc` имеет вид:

```
@rpc(Integer, ViewParams, _returns = ViewInfo)
```

Метод принимает два параметра: целое число (номер сессии) и обязательный параметр типа `ViewParams`, а также возвращает обязательный параметр типа `ViewInfo`, объект которого несёт в себе всю информацию о параметрах метода `core-setup`.

В результате метод `core_setup_view` имеет вид:

```
@rpc(Integer, ViewParams, _returns = ViewInfo)
def core_setup_view (self, sid, params):
```

Параметр `ViewInfo`

Класс `ViewInfo` и все ниже описанные классы (`GroupField`, `Field`) описаны в пакете `calculate-core` в файле `core/server/api_types.py`.

Параметр `ViewInfo` представляет собой массив объектов **`GroupField`** и булевый параметр **`has_brief`**, сообщающий клиенту о том, следует ли выводить информацию о значении всех параметров перед непосредственным запуском метода. Каждый объект **`GroupField`** несёт информацию о группе параметров (например, настройка локализации в методе установки системы) и состоит из:

- `name` - имя группы параметров для отображения;
- `last` - булевое значение, указывает является ли данная группа последний (используется для методов с несколькими шагами);
- `nextlabel` - строка, отображаемая на кнопке, осуществляющей переход на следующий шаг или запуск метода (по умолчанию "Далее" или "Ok", используется для графических клиентов);
- `prevlabel` - строка, отображаемая на кнопке, осуществляющей переход на предыдущий шаг или выход из метода (по умолчанию "Назад" или "Отмена", используется для графических клиентов);
- Массив объектов типа **`Field`**.

Тип параметра `Field`

Объект типа `Field` - описание отдельного параметра, состоящее из полей:

- `name` - имя параметра;
- `label` - отображаемое имя параметра;
- `element` - тип элемента параметра (table, radio, combo, comboEdit, multichoice, input и др.), определяет основные характеристики и правила работы с параметром;
- `type` - тип данных переменной;
- `help` - дополнительная вспомогательная информация о параметре;
- `value` - текущее значение переменной, поступившее от сервера утилит;
- `choice` - список доступных для выбора значений (для элементов radio, combo, comboEdit, multichoice и др.);
- `comments` - список комментариев, соответствующих каждому значению из `choice`, использующийся для отображения пользователю. Как правило, значения из `comments` имеют удобный для человека вид или перевод.
- `listvalue` - список выбранных значений для элементов множественного выбора (multichoice, multichoice_add, selecttable, selecttable_add);
- `tablevalue` - информация о таблице (только для элемента table), состоит из:
 - `head` - заголовок таблиц, определяет название столбцов и ширину таблицы;
 - `body` - тело таблицы, представляет собой данные таблицы, поступающие от сервера утилит;

- **values** - описание типов данных для столбцов (доступные значения, комментарии, доступность на редактирование);
- **onChanged** - зависимость значений столбца от ключевых значений (первого столбца);
- **uncompatible** - булевое значение доступности параметра. Если **uncompatible** равно True, то данный параметр недоступен.
- **default** - состояние параметра, находящегося в расширенных настройках. Определяет, будет ли отображено значение параметра в дополнительных настройках или оно будет пустым (авто).
- **opt** - данные для работы из консоли. Состоит из:
 - **help** - описание параметра (отображается при вызове метода с ключом `-h`, `--help`);
 - **metavalue** - строка, определяющая требуемое значение параметра;
 - **shortopt** - короткий (односимвольный) ключ параметра (например, `-h`);
 - **longopt** - основной ключ параметра (например, `--help`).

Метод процесса

Метод, выполняющий выбранные пользователем действие и запускающийся в отдельном процессе, является обычным методом класса `CoreWSDL`. Для взаимодействия с клиентами (сообщений о прогрессе выполнения процесса, ошибках, успешном выполнении и др.) он использует ряд встроенных функций (методов), описанных в пакете `calculate-core` в файле `core/server/func.py`, таких как:

- **addMessage** - основной метод добавления сообщения в стек сообщений для пользователя (сообщения имеют тип `Message`). Большинство методов являются удобными обёртками для данного метода. Имеет три входных параметра:
 1. **type** - тип передаваемого сообщения;
 2. **message** - текст передаваемое сообщение;
 3. **id** - автоматически генерируемый идентификатор объекта (номер таблицы, прогресса задания и др.);
- **printDefault** - вывод сообщения без отметок
- **printPre** - вывод блока неформатируемого текста
- **printSUCCESS** - обычное сообщение, имеет один параметр **message**(строку сообщения);
- **printWARNING** - сообщение с предупреждением, имеет один параметр **message**(строку сообщения);
- **printERROR** - сообщение об ошибке, имеет один параметр **message**(строку сообщения). Прерывает текущую задачу;
- **startTask** - начало новой задачи, имеет три параметра:
 1. **message** - название задачи;
 2. **progress** - наличие контроля прогресса выполнения задачи (отображение прогрессбара клиентами);
 3. **num** - количество частей для контроля прогресса. Используется только при **progress** равном `True` и определяет, сколько частей от общего прогресса выполнения займёт данная задача. Этот параметр не обязательен, актуален для клиента, работающего в упрощённом режиме вывода с одним прогрессом для всех задач текущего процесса.
- **setTaskNumber** - установка числа частей для задач. Вызывается перед первым вызовом метода **startTask**. Если в вызовах **startTask** не указывается количество частей (параметр **num**), то в вызове **setTaskNumber** нет необходимости. Имеет один целочисленный параметр - число частей для задач, которое должно быть равно сумме всех значений параметра **num**, указанных в методах **startTask**;
- **endTask** - сообщение о завершении текущей задачи. Имеет два параметра:
 1. **result** - сообщение о завершении задачи;
 2. **progress_message** - сообщение, используемое при выводе прогресса;
- **endFrame** - сообщение пользователю о завершении работы процесса;
- **addProgress** - добавление прогресса для задания;

- `setProgress` - установить процент выполнения для текущего прогресса. Имеет три входных параметра:
 1. `perc` - процент выполнения текущей задачи;
 2. `short_message` - короткое сообщение для текущего процента выполнения;
 3. `long_message` - длинное сообщение для текущего процента выполнения;
- `getProgress` - получение текущего прогресса (процента) выполнения задачи;
- `printTable` - отображение таблицы. Имеет шесть параметров (три основных и три, использующихся при необходимости взаимодействия пользователя с таблицей в графических клиентах):
 1. `table_name` - отображаемое название таблицы;
 2. `head` - заголовок таблицы;
 3. `body` - тело таблицы;
 4. `onClick` - название метода, вызываемого при нажатии на строку таблицы. При этом будут переданы параметры из этой строки, описанные в поле `fields` (см. далее);
 5. `fields` - список полей, необходимых для передачи в вызываемый метод, имя которого хранится в параметре `onClick`. Список `fields` равен по длине заголовку таблицы, неиспользуемые для передачи поля равны пустым строкам. Например, при `fields = ['cl_group_name', "", "]` в вызываемый метод передастся параметр `cl_group_name`, значение для которого будет взято из первого столбца выбранной строки;
 6. `addAction` - добавить кнопку добавления над таблицей (кнопка с иконкой "плюс"), при нажатии на которую будет вызван метод, имя которого указано в параметре `addAction`;
- `setStatus` - установить статус процесса. 0 - процесс успешно завершён, 1 - процесс выполняется, 2 - процесс завершён с ошибкой;
- `getStatus` - получить текущий статус процесса;
- `askQuestion` - получить от пользователя дополнительные данные (задать вопрос пользователю). Имеет один параметр `message` - текст сообщения. Вызов этого метода возвратит введённое пользователем значение;
- `askPassword` - получить пароль от пользователя. Имеет два параметра: `message` (текст сообщения) и `twice` (необходимость дублирования ввода пароля).
- `writeFile` - записать текущее состояние процесса в файл (по умолчанию `/var/calculate/server/pids/N.pid`, где N - идентификатор процесса (pid)).
- `askChoice` - получить от пользователя выбор из представленных вариантов
- `askConfirm` - получить от пользователя подтверждение
- `isInteractive` - узнать интерактивно ли запущена действие в консоли (действие может из консоли может быть запущено с ключом `-f`)

Метод должен вернуть значение `True` при успешном завершении и значение `False` в случае ошибки.

Категория Установка

Для установки системы необходим установленный пакет `calculate-install`. В категории **Установка** находятся три метода (при наличии необходимых прав у сертификата):

- Установка системы (`install`);
- Установка на Flash (`install_flash`);
- Установка [PXE](#) (`install_pxe`).

Все методы установки системы работают через общий метод `installCommon` и описаны в файле `install/cl_wsdl_install.py` пакета `calculate-install` (`/usr/lib/python2.7/site-packages/calculate/install/cl_wsdl_install.py`).

Установка системы

Установка системы включает в себя:

- **install** - гурс метод, проверяющий входные параметры и запускающий отдельный процесс установки системы;

Входные данные метода **install**:

1. sid - идентификатор (номер) сессии;
2. info - объект типа InstallInfo, хранящий в себе все параметры, необходимые для установки системы (класс InstallInfo объявлен в файле install/cl_wsdl_install.py пакета calculate-install (/usr/lib/python2.7/site-packages/calculate/install/cl_wsdl_install.py)).

Результирующие данные метода **install** - список объектов типа ReturnedMessage, несущих информацию о запуске процесса (идентификатор процесса pid) или сообщения с ошибками.

- **install_view** - гурс метод, создающий объект ViewInfo с данными о параметрах установки и передающий его клиенту;

Входные данные метода **install_view**:

1. sid - идентификатор (номер) сессии;
2. params - объект типа ViewParams (описание типа ViewParams см. выше в разделе "Метод, предоставляющий дополнительную информацию").

Результирующие данные метода **install_view** - объект типа ViewInfo, несущий информацию о всех параметрах установки, допустимых значениях и способах отображения элементов (описание типа ViewInfo см. выше в разделе "Метод, предоставляющий дополнительную информацию").

- **install_vars** - вспомогательный метод, формирующий структуру объекта ViewInfo;
- **Install** - основной класс установки и настройки системы.

Установка системы на Flash

Установка системы на Flash включает в себя:

- **install_flash** - гурс метод, проверяющий входные параметры и запускающий отдельный процесс установки системы на Flash носитель;

Входные данные метода **install_flash** аналогичны входным данным метода **install**:

1. sid - идентификатор (номер) сессии;
2. info - объект типа InstallInfo, хранящий в себе все параметры, необходимые для установки системы.

Результирующие данные метода **install_flash** - список объектов типа ReturnedMessage, несущих информацию о запуске процесса (идентификатор процесса pid) или сообщения с ошибками.

- **install_flash_view** - гурс метод, создающий объект ViewInfo с данными о параметрах установки и передающий его клиенту;

Входные данные метода **install_flash_view**:

1. sid - идентификатор (номер) сессии;
2. params - объект типа ViewParams (описание типа ViewParams см. выше в разделе "Метод, предоставляющий дополнительную информацию").

Результирующие данные метода **install_flash_view** - объект типа ViewInfo, несущий информацию о всех параметрах установки, допустимых значениях и способах отображения

элементов.

- **install_flash_vars** - вспомогательный метод, формирующий структуру объекта ViewInfo для методов install_flash и install_flash_view;
- **Install** - основной класс установки и настройки системы.

Установка PXE

Установка PXE включает в себя:

- **install_pxe** - grpc метод, проверяющий входные параметры и запускающий отдельный процесс установки системы с применением [PXE](#) (Установка PXE доступна только на Calculate Directory Server);

Входные данные метода **install_pxe** аналогичны входным данным метода **install**:

1. sid - идентификатор (номер) сессии;
2. info - объект типа InstallInfo, хранящий в себе все параметры, необходимые для установки системы.

Результирующие данные метода **install_pxe** - список объектов типа ReturnedMessage, несущих информацию о запуске процесса (идентификатор процесса pid) или сообщения с ошибками.

- **install_pxe_view** - grpc метод, создающий объект ViewInfo с данными о параметрах установки и передающий его клиенту;

Входные данные метода **install_pxe_view**:

1. sid - идентификатор (номер) сессии;
2. params - объект типа ViewParams (описание типа ViewParams см. выше в разделе "Метод, предоставляющий дополнительную информацию").

Результирующие данные метода **install_pxe_view** - объект типа ViewInfo, несущий информацию о всех параметрах установки, допустимых значениях и способах отображения элементов.

- **install_flash_vars** - вспомогательный метод, формирующий структуру объекта ViewInfo для методов install_flash и install_flash_view;
- **Install** - основной класс установки и настройки системы.

Категория Настройка

К категории **Настройка** относятся:

- Локализация (setup_locale);
- Загрузка (setup_boot);
- Сеть (setup_network);
- Видео (setup_video);
- Настройка пакета (core_setup);
- Система (setup_system).

Локализация

Настройка локализации, заключающаяся в настройки языка и часового пояса, включает в себя:

- **setup_locale** - grpc метод, проверяющий входные параметры и запускающий отдельный процесс настройки локализации;

Входные данные метода **setup_locale**:

1. sid - идентификатор (номер) сессии;
2. info - объект типа InstallInfo, хранящий в себе все параметры, необходимые для установки и настройки системы.

Результирующие данные метода **setup_locale** - список объектов типа ReturnedMessage, несущих информацию о запуске процесса (идентификатор процесса pid) или сообщения с ошибками.

- **setup_locale_view** - гrpc метод, создающий объект ViewInfo с данными о параметрах настройки и передающий его клиенту;

Входные данные метода **setup_locale_view**:

1. sid - идентификатор (номер) сессии;
2. params - объект типа ViewParams (описание типа ViewParams см. выше в разделе "Метод, предоставляющий дополнительную информацию").

Результирующие данные метода **setup_locale_view** - объект типа ViewInfo, несущий информацию о всех параметрах настройки локализации, допустимых значениях и способах отображения элементов (описание типа ViewInfo см. выше в разделе "Метод, предоставляющий дополнительную информацию").

- **setup_locale_vars** - вспомогательный метод, формирующий структуру объекта ViewInfo для настройки локализации;
- **Install** - основной класс установки и настройки системы.

Загрузка

Настройка загрузки, заключающаяся в изменении конфигурации [grub](#), включает в себя:

- **setup_boot** - гrpc метод, проверяющий входные параметры и запускающий отдельный процесс настройки загрузки;

Входные данные метода **setup_boot**:

1. sid - идентификатор (номер) сессии;
2. info - объект типа InstallInfo, хранящий в себе все параметры, необходимые для установки и настройки системы.

Результирующие данные метода **setup_boot** - список объектов типа ReturnedMessage, несущих информацию о запуске процесса (идентификатор процесса pid) или сообщения с ошибками.

- **setup_boot_view** - гrpc метод, создающий объект ViewInfo с данными о параметрах настройки и передающий его клиенту;

Входные данные метода **setup_boot_view**:

1. sid - идентификатор (номер) сессии;
2. params - объект типа ViewParams (описание типа ViewParams см. выше в разделе "Метод, предоставляющий дополнительную информацию").

Результирующие данные метода **setup_boot_view** - объект типа ViewInfo, несущий информацию о всех параметрах настройки загрузки, допустимых значениях и способах отображения элементов (описание типа ViewInfo см. выше в разделе "Метод,

предоставляющий дополнительную информацию").

- **setup_boot_vars** - вспомогательный метод, формирующий структуру объекта ViewInfo для настройки загрузки;
- **Install** - основной класс установки и настройки системы.

Сеть

Настройка сети заключается в выборе менеджера сети, настройке сетевых интерфейсов, указании имени хоста, сервера времени, сервера доменных имён, доменов для поиска и таблицы маршрутизации. Настройка сети включает в себя:

- **setup_network** - grpc метод, проверяющий входные параметры и запускающий отдельный процесс настройки сети;

Входные данные метода **setup_network**:

1. sid - идентификатор (номер) сессии;
2. info - объект типа InstallInfo, хранящий в себе все параметры, необходимые для установки и настройки системы.

Результирующие данные метода **setup_network** - список объектов типа ReturnedMessage, несущих информацию о запуске процесса (идентификатор процесса pid) или сообщения с ошибками.

- **setup_network_view** - grpc метод, создающий объект ViewInfo с данными о параметрах настройки и передающий его клиенту;

Входные данные метода **setup_network_view**:

1. sid - идентификатор (номер) сессии;
2. params - объект типа ViewParams (описание типа ViewParams см. выше в разделе "Метод, предоставляющий дополнительную информацию").

Результирующие данные метода **setup_network_view** - объект типа ViewInfo, несущий информацию о всех параметрах настройки сети, допустимых значениях и способах отображения элементов (описание типа ViewInfo см. выше в разделе "Метод, предоставляющий дополнительную информацию").

- **setup_network_vars** - вспомогательный метод, формирующий структуру объекта ViewInfo для настройки сети;
- **Install** - основной класс установки и настройки системы.

Видео

Настройка видео заключается в выборе видео драйвера, разрешения экрана, разрешения фреймбуфера и установке композита.

Настройка видео включает в себя:

- **setup_video** - grpc метод, проверяющий входные параметры и запускающий отдельный процесс настройки видео;

Входные данные метода **setup_video**:

1. sid - идентификатор (номер) сессии;
2. info - объект типа InstallInfo, хранящий в себе все параметры, необходимые для установки и настройки системы.

Результирующие данные метода **setup_video** - список объектов типа ReturnedMessage, несущих информацию о запуске процесса (идентификатор процесса pid) или сообщения с ошибками.

- **setup_video_view** - grpc метод, создающий объект ViewInfo с данными о параметрах настройки и передающий его клиенту;

Входные данные метода **setup_video_view**:

1. sid - идентификатор (номер) сессии;
2. params - объект типа ViewParams (описание типа ViewParams см. выше в разделе "Метод, предоставляющий дополнительную информацию").

Результирующие данные метода **setup_video_view** - объект типа ViewInfo, несущий информацию о всех параметрах настройки видео, допустимых значениях и способах отображения элементов (описание типа ViewInfo см. выше в разделе "Метод, предоставляющий дополнительную информацию").

- **setup_video_vars** - вспомогательный метод, формирующий структуру объекта ViewInfo для настройки видео;
- **Install** - основной класс установки и настройки системы.

Система

Настройка системы заключается в наложении системных шаблонов (локальных, удалённых и шаблонов из overlay).

Настройка системы включает в себя:

- **setup_system** - grpc метод, проверяющий входные параметры и запускающий отдельный процесс настройки системы;

Входные данные метода **setup_system**:

1. sid - идентификатор (номер) сессии;
2. info - объект типа InstallInfo, хранящий в себе все параметры, необходимые для установки и настройки системы.

Результирующие данные метода **setup_system** - список объектов типа ReturnedMessage, несущих информацию о запуске процесса (идентификатор процесса pid) или сообщения с ошибками.

- **setup_system_view** - grpc метод, создающий объект ViewInfo с данными о параметрах настройки и передающий его клиенту;

Входные данные метода **setup_system_view**:

1. sid - идентификатор (номер) сессии;
2. params - объект типа ViewParams (описание типа ViewParams см. выше в разделе "Метод, предоставляющий дополнительную информацию").

Результирующие данные метода **setup_system_view** - объект типа ViewInfo, несущий информацию о всех параметрах настройки системы, допустимых значениях и способах отображения элементов (описание типа ViewInfo см. выше в разделе "Метод, предоставляющий дополнительную информацию").

- **setup_system_vars** - вспомогательный метод, формирующий структуру объекта ViewInfo для настройки системы;

- **Install** - основной класс установки и настройки системы.

Настройка пакета

Настройка пакета заключается в выборе пакета и наложении шаблонов для него.

Настройка пакета включает в себя:

- **core_setup** - гrpc метод, проверяющий входные параметры и запускающий отдельный процесс настройки пакета;

Входные данные метода **setup_video**:

1. sid - идентификатор (номер) сессии;
2. info - объект типа CoreSetupInfo, хранящий в себе все параметры, необходимые для настройки пакета (класс CoreSetupInfo объявлен в файле core/server/setup_package.py пакета calculate-core).

Результирующие данные метода **setup_video** - список объектов типа ReturnedMessage, несущих информацию о запуске процесса (идентификатор процесса pid) или сообщения с ошибками.

- **core_setup_view** - гrpc метод, создающий объект ViewInfo с данными о параметрах настройки и передающий его клиенту;

Входные данные метода **setup_video_view**:

1. sid - идентификатор (номер) сессии;
2. params - объект типа ViewParams (описание типа ViewParams см. выше в разделе "Метод, предоставляющий дополнительную информацию").

Результирующие данные метода **setup_video_view** - объект типа ViewInfo, несущий информацию о всех параметрах настройки пакета, допустимых значениях и способах отображения элементов (описание типа ViewInfo см. выше в разделе "Метод, предоставляющий дополнительную информацию").

- **core_setup_vars** - вспомогательный метод, формирующий структуру объекта ViewInfo для настройки пакета;
- **updateSystemConfigs** - основной класс настройки пакета.

Категория Утилиты

Категория Утилиты состоит из трёх групп методов:

- Управление запросами (core_show_request, core_detail_request, core_confirm_request, core_del_request);
- Просмотр сертификатов (core_view_cert);
- Управление группами (core_show_groups, core_detail_group, core_add_group, core_change_group, core_del_group).

Управление запросами

Группа методов управления запросами состоит из методов просмотра, подробного просмотра, подтверждения (подписания) и удаления запросов.

Просмотр запросов выводит таблицу текущих запросов на сервере утилит, включает в себя:

- **core_show_request** - гrpc метод, проверяющий входные параметры и вызывающий вспомогательный метод requestCommon, который запускает процесс формирования таблицы запросов на сервере утилит;

Входные данные метода **core_show_request**:

1. sid - идентификатор (номер) сессии;
2. info - объект типа RequestInfo, хранящий в себе параметры выдаваемой таблицы запросов (количество строк и смещение). Класс RequestInfo объявлен в файле core/server/request.py пакета calculate-core.

Результирующие данные метода **core_show_request** - список объектов типа ReturnedMessage, несущих информацию о запуске процесса (идентификатор процесса pid) или сообщения с ошибками.

- **core_show_request_view** - гrpc метод, создающий объект ViewInfo с данными, необходимыми для формирования таблицы запросов, и передающий его клиенту;

Входные данные метода **core_show_request_view**:

1. sid - идентификатор (номер) сессии;
2. params - объект типа ViewParams (описание типа ViewParams см. выше в разделе "Метод, предоставляющий дополнительную информацию").

Результирующие данные метода **core_show_request_view** - объект типа ViewInfo, несущий информацию о параметрах таблицы запросов (описание типа ViewInfo см. выше в разделе "Метод, предоставляющий дополнительную информацию").

- **show_request_meth** - метод, формирующий таблицу запросов на сервере утилит;
- **requestCommon** - метод, выполняющий дополнительные проверки и запускающий процесс формирования таблицы запросов на сервере утилит.

Подробный просмотр запросов реализован для графических клиентов и осуществляет взаимодействие метода просмотра запросов с методами подтверждения и удаления запросов на сервере утилит, включает в себя:

- **core_detail_request** - гrpc метод, проверяющий входные параметры (номер запроса). Не вызывает никаких других методов (является заглушкой);

Входные данные метода **core_detail_request**:

1. sid - идентификатор (номер) сессии;
2. info - объект типа DetailRequestInfo, хранящий в себе параметры запроса (номер и имя группы для подписания сертификата). Класс DetailRequestInfo объявлен в файле core/server/request.py пакета calculate-core.

Результирующие данные метода **core_detail_request** - список объектов типа ReturnedMessage, несущих информацию о запуске процесса (идентификатор процесса pid) или сообщения с ошибками.

- **core_detail_request_view** - гrpc метод, создающий объект ViewInfo с данными запроса. Содержит данные о запросе и кнопки вызова других методов (просмотра, подписания и удаления запросов, с передачей необходимых параметров);

Входные данные метода **core_detail_request_view**:

1. sid - идентификатор (номер) сессии;
2. params - объект типа ViewParams (описание типа ViewParams см. выше в разделе "Метод, предоставляющий дополнительную информацию").

Результирующие данные метода **core_detail_request_view** - объект типа ViewInfo, несущий информацию о всех параметрах запроса, допустимых значениях и способах отображения элементов, а так же кнопках перехода к другим методам и передачи им параметров (описание

типа ViewInfo см. выше в разделе "Метод, предоставляющий дополнительную информацию").

Подтверждение (подписание) запросов осуществляет создание сертификата клиента, включает в себя:

- **core_confirm_request** - grpc метод, проверяющий входные параметры и вызывающий вспомогательный метод confirmRequestCommon, который запускает процесс подписания запроса на сертификат;

Входные данные метода **core_confirm_request**:

1. sid - идентификатор (номер) сессии;
2. info - объект типа DetailRequestInfo, хранящий в себе номер запроса и имя группы для подписания сертификата.

Результирующие данные метода **core_confirm_request** - список объектов типа ReturnedMessage, несущих информацию о запуске процесса (идентификатор процесса pid) или сообщения с ошибками.

- **core_confirm_request_view** - grpc метод, создающий объект ViewInfo с данными, необходимыми для формирования таблицы запросов, и передающий его клиенту;

Входные данные метода **core_confirm_request_view**:

1. sid - идентификатор (номер) сессии;
2. params - объект типа ViewParams (описание типа ViewParams см. выше в разделе "Метод, предоставляющий дополнительную информацию").

Результирующие данные метода **core_confirm_request_view** - объект типа ViewInfo, несущий информацию о параметрах запроса (номер запроса и имя группы для подписания), допустимых значениях и способах отображения элементов (описание типа ViewInfo см. выше в разделе "Метод, предоставляющий дополнительную информацию").

- **confirm_request_meth** - метод, выполняющий подписание сертификата;
- **confirmRequestCommon** - метод, выполняющий дополнительные проверки и запускающий процесс подписания сертификата (confirm_request_meth).

Удаление запроса включает в себя:

- **core_del_request** - grpc метод, проверяющий входные параметры и вызывающий вспомогательный метод delRequestCommon, который удаляет все данные о запросе с сервера утилит;

Входные данные метода **core_del_request**:

1. sid - идентификатор (номер) сессии;
2. info - объект типа DetailRequestInfo, хранящий в себе номер запроса для его удаления.

Результирующие данные метода **core_del_request** - список объектов типа ReturnedMessage, несущих информацию о запуске процесса (идентификатор процесса pid) или сообщения с ошибками.

- **core_del_request_view** - grpc метод, создающий объект ViewInfo с данными, необходимыми для удаления запроса, и передающий его клиенту;

Входные данные метода **core_del_request_view**:

1. sid - идентификатор (номер) сессии;

2. params - объект типа ViewParams (описание типа ViewParams см. выше в разделе "Метод, предоставляющий дополнительную информацию").

Результирующие данные метода **core_del_request_view** - объект типа ViewInfo, несущий информацию о параметрах запроса (номер запроса), допустимых значениях и способах отображения элементов (описание типа ViewInfo см. выше в разделе "Метод, предоставляющий дополнительную информацию").

- **del_request_meth** - метод, выполняющий удаления запроса;
- **delRequestCommon** - метод, выполняющий дополнительные проверки и запускающий процесс удаления запроса (del_request_meth).

Просмотр сертификатов

Просмотр сертификатов заключается в просмотре данных по выбранному сертификату: доступных методов, группы прав, данных о подписчике и субъекте, серийного номера и отпечатка сертификата. Просмотр сертификатов включает в себя:

- **core_view_cert** - гурс метод, проверяющий входные параметры и вызывающий вспомогательный метод certificateCommon;

Входные данные метода **core_view_cert**:

1. sid - идентификатор (номер) сессии;
2. info - объект типа CertificateInfo, хранящий в себе номер сертификата.

Результирующие данные метода **core_view_cert** - список объектов типа ReturnedMessage, несущих информацию о запуске процесса (идентификатор процесса pid) или сообщения с ошибками.

- **core_view_cert_view** - гурс метод, создающий объект ViewInfo с данными, необходимыми выбора сертификата для просмотра информации о нём, и передающий его клиенту;

Входные данные метода **core_view_cert_view**:

1. sid - идентификатор (номер) сессии;
2. params - объект типа ViewParams (описание типа ViewParams см. выше в разделе "Метод, предоставляющий дополнительную информацию").

Результирующие данные метода **core_view_cert_view** - объект типа ViewInfo, несущий информацию о параметрах (номера сертификатов), допустимых значениях и способах отображения элементов (описание типа ViewInfo см. выше в разделе "Метод, предоставляющий дополнительную информацию").

- **view_cert_meth** - метод, выполняющий сбор информации о сертификате (доступных для выполнения методов, группы прав сертификата, данных о подписчике и субъекте, серийного номера и отпечатка сертификата);
- **certificateCommon** - метод, выполняющий дополнительные проверки и запускающий процесс просмотра данных о сертификате (view_cert_meth).

Управление группами

Группа методов управления группами состоит из методов просмотра, подробного просмотра, добавления, изменения и удаления групп прав сертификатов.

Просмотр групп выводит таблицу текущих групп на сервере утилит, включает в себя:

- **core_show_groups** - grpc метод, проверяющий входные параметры и вызывающий вспомогательный метод groupCommon, который запускает процесс формирования таблицы групп, существующих на сервере утилит;

Входные данные метода **core_show_groups**:

1. sid - идентификатор (номер) сессии;
2. info - объект типа GroupInfo, хранящий в себе параметры выдаваемой таблицы запросов (количество строк и смещение). Класс GroupInfo объявлен в файле core/server/edit_groups.py пакета calculate-core.

Результирующие данные метода **core_show_groups** - список объектов типа ReturnedMessage, несущих информацию о запуске процесса (идентификатор процесса pid) или сообщения с ошибками.

- **core_show_groups_view** - grpc метод, создающий объект ViewInfo с данными, необходимыми для формирования таблицы групп, и передающий этот объект клиенту;

Входные данные метода **core_show_groups_view**:

1. sid - идентификатор (номер) сессии;
2. params - объект типа ViewParams (описание типа ViewParams см. выше в разделе "Метод, предоставляющий дополнительную информацию").

Результирующие данные метода **core_show_groups_view** - объект типа ViewInfo, несущий информацию о параметрах таблицы группы (описание типа ViewInfo см. выше в разделе "Метод, предоставляющий дополнительную информацию").

- **show_groups_meth** - метод, формирующий таблицу групп на сервере утилит;
- **groupCommon** - метод, выполняющий дополнительные проверки и запускающий процесс формирования таблицы групп на сервере утилит (запуск метода show_groups_meth).

Подробный просмотр группы реализован для графических клиентов и осуществляет взаимодействие метода просмотра групп с методами изменения и удаления групп на сервере утилит, включает в себя:

- **core_detail_group** - grpc метод, проверяющий входные параметры (имя группы). Не вызывает никаких других методов (является заглушкой);

Входные данные метода **core_detail_group**:

1. sid - идентификатор (номер) сессии;
2. info - объект типа DetailGroupInfo, хранящий в себе параметры группы (имя группы и список прав). Класс DetailGroupInfo объявлен в файле core/server/edit_groups.py пакета calculate-core.

Результирующие данные метода **core_detail_group** - список объектов типа ReturnedMessage, несущих информацию о запуске процесса (идентификатор процесса pid) или сообщения с ошибками.

- **core_detail_group_view** - grpc метод, создающий объект ViewInfo с данными группы. Содержит данные о группе и кнопки вызова других методов (просмотра, изменения и удаления группы, с передачей необходимых параметров);

Входные данные метода **core_detail_group_view**:

1. sid - идентификатор (номер) сессии;
2. params - объект типа ViewParams (описание типа ViewParams см. выше в разделе "Метод, предоставляющий дополнительную информацию").

Результирующие данные метода **core_detail_group_view** - объект типа ViewInfo, несущий информацию о параметрах группы прав, допустимых значениях и способах отображения элементов, а так же кнопках перехода к другим методам и передачи им параметров (описание типа ViewInfo см. выше в разделе "Метод, предоставляющий дополнительную информацию").

Добавление группы создаёт новую группу прав на сервере утилит (для графического клиента - кнопка плюс "+" над таблицей групп). Включает в себя:

- **core_add_group** - грс метод, проверяющий входные параметры и вызывающий вспомогательный метод addGroupCommon;

Входные данные метода **core_add_group**:

1. sid - идентификатор (номер) сессии;
2. info - объект типа AddGroupInfo, хранящий в себе имя новой группы и список прав для неё.

Результирующие данные метода **core_add_group** - список объектов типа ReturnedMessage, несущих информацию о запуске процесса (идентификатор процесса pid) или сообщения с ошибками.

- **core_add_group_view** - грс метод, создающий объект ViewInfo с данными, необходимыми для добавления группы, и передающий его клиенту;

Входные данные метода **core_add_group_view**:

1. sid - идентификатор (номер) сессии;
2. params - объект типа ViewParams (описание типа ViewParams см. выше в разделе "Метод, предоставляющий дополнительную информацию").

Результирующие данные метода **core_add_group_view** - объект типа ViewInfo, несущий информацию о параметрах новой группы (имя и список прав), допустимых значениях и способах отображения элементов (описание типа ViewInfo см. выше в разделе "Метод, предоставляющий дополнительную информацию").

- **add_group_meth** - метод, выполняющий добавление новой группы;
- **addGroupCommon** - метод, выполняющий дополнительные проверки и запускающий процесс добавления группы (add_group_meth).

Вспомогательные модули

Вспомогательные модули служат для обеспечения взаимодействия клиента и сервера утилит. По выполняемым функциям методы этих модулей можно разделить на пять типов:

- работа с сертификатами, запросами и списками отзывов сертификатов;
- работа с сессиями;
- работа с процессами;
- работа с кэшем сессий, методов и процессов;
- прочие методы.

Методы работы с сертификатами

К вспомогательным методам работы с сертификатами относятся:

- **post_client_request** - отправить клиентский запрос на подпись сертификата. Вызов этого метода передаёт на сервер утилит от клиента запрос на подпись сертификата.

Имеет четыре входных параметра:

- **request** - запрос на подпись сертификата (в виде строки с содержимым запроса);
- **ip** - IP-адрес (уникальный сетевой адрес узла в компьютерной сети);
- **mac** - MAC-адрес (уникальный идентификатор, присваиваемый каждой единице оборудования компьютерных сетей);
- **client_type** - тип клиента (console или gui).

Возвращает номер запроса или значение "-1" в случае ошибки.

- **post_server_request** - отправить серверный запрос на подпись сертификата. Вызов этого метода передаёт на сервер утилит запрос на подпись сертификата от другого сервера утилит.

Имеет три входных параметра:

- **request** - запрос на подпись сертификата (в виде строки с содержимым запроса);
- **ip** - IP-адрес;
- **mac** - MAC-адрес;

Возвращает номер запроса или значение "-1" в случае ошибки.

- **get_client_cert** - забрать подписанный сертификат клиента с сервера утилит.

Имеет два входных параметра:

- **req_id** - номер запроса (уникальный номер запроса на сервере утилит, выдаётся при вызове метода `post_client_request`).
- **request** - запрос на подпись сертификата. Необходим для проверки с ранее сохранённым.

Возвращает значение "1", если запрос был отвергнут или не найден на сервере, значение "2", если сертификата не существует (запрос ещё не был подписан). В случае подписания возвращает список из двух сертификатов - сертификат клиента и корневой сертификат сервера.

- **get_server_cert** - забрать подписанный сертификат сервера утилит.

Имеет аналогичные с методом `get_client_cert` входные и возвращаемые параметры (см. выше).

- **get_ca** - получить корневой сертификат сервера утилит.

возвращает "1", если в сертификате сервера утилит нет поля "CN", значение "2", если не найден сертификат, которым подписан сертификат сервера, иначе возвращает сертификат, которым подписан сертификат сервера.

- **get_crl** - получить список отзыва сертификатов.

Возвращает содержимое списка отзывов сертификатов или значение " ".

- **post_cert** - проверка сертификата на сервере. Вызывается при установке соединения для проверки наличия сертификата клиента на сервере и проверки сертификата на срок годности. Используется для графических клиентов, для консольных используется метод `init_session`.

Не имеет входных параметров. Возвращает список целочисленных значений: номер сертификата и значение годности сертификата (принимает значения: "-2" - срок годности истёк, "-1" - срок годности более 2-х месяцев или количество дней до истечения срока годности). Если у клиента нет сертификата возвращает значение [-3], если сертификат клиента не найден на сервере - значение [-4].

Методы работы с сессиями

К вспомогательным методам работы с сессиями относятся:

- **post_sid** - получение номера сессии с сервера утилит или продолжение незакрытой предыдущей сессии. Также вызов метода устанавливает текущий язык для локализации. Используется графическими клиентами.

Имеет три входных параметра:

- **sid** - номер сессии, хранящийся у клиента. В случае, если сессия не была закрыта, она будет продолжена;
- **cert_id** - номер сертификата;
- **lang** - текущий язык клиента.

Возвращает список из двух целочисленных значений: номера сессии и признака продолжения старой сессии (1 - новая сессия, 0 - старая сессия).

- **del_sid** - закрытие (удаление) удаление сессии на сервере утилит.

Имеет единственный входной параметр **sid** - идентификатор (номер) сессии.

Возвращает значение ['0'] при успешном закрытии сессии.

- **get_sessions** - получения списка текущих сессий на сервере утилит.

Имеет один входной параметр **sid** - номер своей сессии, необходимый для проверки соответствия сертификату.

Возвращает значение [""], если сессия не соответствует сертификату, иначе список всех активных сессий на сервере утилит.

- **sid_info** - получить информацию по выбранной сессии.

Имеет один входной параметр **sid** - номер сессии.

Возвращает список значений: номер сертификата, дату подписания сертификата, ip-адрес, mac-адрес и тип клиента при указанные при генерации сертификата, а также значение, активна ли сессия ('0' - активна, иначе - нет).

- **init_session** - установка начального соединения консольного клиента. Выполняет проверку сертификата клиента и получает номер сессии (или продолжает незакрытую сессию).

Аналогичен действиям методов **post_sid** и **post_cert** для графических клиентов.

Имеет два входных параметра:

- **sid** - номер сессии, хранящийся у клиента. В случае, если сессия не была закрыта, она будет продолжена;
- **lang** - текущий язык клиента.

Возвращает кортеж (список) из двух списков:

- первый - список целочисленных значений: номер сертификата и значение годности сертификата (принимает значения: "-2" - срок годности истёк, "-1" - срок годности более 2-х месяцев или количество дней до истечении срока годности). Если у клиента нет сертификата возвращает значение [-3], если сертификат клиента не найден на сервере - значение [-4].
- второй список из двух целочисленных значений: номера сессии и признака

продолжения старой сессии (1 - новая сессия, 0 - старая сессия).

Методы работы с процессами

К вспомогательным методам работы с процессами относятся:

- **list_pid** - получить список запущенных процессов (а так же завершивших работу, но принадлежащих данной сессии).

Имеет один входной параметр **sid** - номер текущей сессии.

Возвращает список номеров запущенных процессов или значение [0].

- **pid_info** - информация по текущему процессу.

Имеет два входных параметра:

- **sid** - номер текущей сессии;
- **pid** - номер процесса, по которому необходимо просмотреть информацию.

Возвращает список значений: идентификатор (номер) процесса, статус процесса ('1' - активен, '0' - завершён, иначе прерван или завершён с ошибкой), время запуска процесса, имя процесса (запустившей его функции), имя груп метода процесса.

pid_kill - завершить (прервать) выполняющийся процесс.

Имеет два параметра, аналогичных методу pid_info.

Возвращает значение -2, если сессия не соответствует сертификату, значение 3, если нет такого процесса на сервере утилит, значение 1, если не удалось послать процессу сигнал завершения и значение 0, если процессу послан сигнал завершения.

Для получения информации от выполняющихся процессов используются следующие методы: **get_entire_frame**, **get_frame**, **get_progress**, **get_table** и **send_message**. Методы **get_entire_frame**, **get_frame** и **send_message** возвращают данные типа **Message** (или массив таких данных). Возвращаемый сервером тип **Message** служит для сообщения клиенту информации о работе запущенного процесса. Добавление таких сообщений процессом описано ранее в разделе "Основные модули" -> "Метод процесса". Каждый объект типа **Message** состоит из следующих полей:

1. **type** - тип сообщения ("error", "warning", "table", "startTask" и другие);
2. **message** - передаваемое сообщение;
3. **id** - уникальный номер (идентификатор), служащий для указания номера таблицы, номера значения прогресса или количества частей всех заданий и количество частей для каждого задания.
4. **get_entire_frame** - получить список всех сообщений процесса.

Имеет два входных параметра:

- **sid** - номер сессии;
- **pid** - номер выполняющегося процесса.

Возвращает массив объектов типа **Message**.

- **get_frame** - получить список новых (неполученных ранее) сообщений процесса.
Входные и возвращаемые значения аналогичны параметрам метода **get_entire_frame**.
- **get_progress** - получение значения прогресса для текущей задачи.

Имеет три входных параметра:

- **sid** - номер сессии;
- **pid** - номер процесса;
- **id** - идентификатор прогресса задачи.

Возвращает объект типа ReturnProgress, состоящий из полей:

- **percent** - целочисленное значение прогресса (процент);
- **short_message** - короткое сообщение для текущего прогресса;
- **long_message** - полное сообщение для текущего прогресса.

- **get_table** - получить таблицу с сервера утилит.

Имеет аналогичные с методом get_progress параметры, где **id** - идентификатор таблицы для процесса.

Возвращает объект типа Table, включающий в себя поля:

- **head** - шапка таблицы (список строк);
 - **onClick** - имя метода, вызываемого при нажатии на строку таблицы.
 - **fields** - имена полей (список строк), для передачи в вызываемый метод, имя которого указано в поле onClick. Номер поля в списке соответствует номеру столбца, из которого будет присвоено значение. Указывается только совместно с полем onClick;
 - **body** - тело таблицы (двумерный массив строк);
 - **addAction** - имя метода, отвечающего за добавление строки таблицы. Поле используется для графических клиентов, добавляет кнопку плюс "+" над таблицей, при нажатии на которую вызывается метод.
 - **values** - список объектов типа ChoiceValue, в методах printTable и get_table не используется (поле используется только в таблицах, передаваемых в составе объекта ViewInfo);
-
- **send_message** - Отослать сообщение на сервер утилит. Посыпает ответ на запросы процесса, осуществляемые вызовом методов askQuestion и askPassword.

Имеет три входных параметра:

- **sid** - номер сессии;
- **pid** - номер процесса;
- **text** - текст ответа.

Возвращает объект типа Message

Методы работы с кэшем

Для работы с [кэшем](#) существует три метода, позволяющих на разных уровнях очищать текущие данные:

- **clear_session_cache** - очищает кэш для всей сессии.

Имеет один параметр **sid** - номер сессии.

Возвращает 1 в случае ошибки и 0 при успешной очистке кэша сессии.

- **clear_method_cache** - очищает кэш метода, принадлежащего данной сессии. Кэш метода создаётся до запуска процесса и хранит значение параметров, передаваемых от клиента серверу утилит.

Метод `clear_method_cache` имеет два входных параметра:

- `sid` - номер сессии.
- `method_name` - имя метода.

Возвращает 1 в случае ошибки и 0 при успешной очистке кэша метода.

- `clear_pid_cache` - очищает кэш процесса. После завершения работы процесс хранит все данные о своей работе, чтобы пользователь мог просмотреть результаты.

Метод `clear_pid_cache` имеет два входных параметра:

- `sid` - номер сессии;
- `pid` - номер процесса.

Возвращает значение 1 в случае ошибки, значение 2 при отсутствии номера процесса в списке номеров процессов сессии и 0 при успешной очистке кэша процесса.

Прочие вспомогательные методы

К методам, не попавшим не в одну из предыдущих групп относятся:

- `get_methods` - получение всех доступных (по правам сертификата) методов на сервере утилит.

Имеет два входных параметра:

- `sid` - номер сессии;
- `client_type` - тип клиента (gui или console).

Метод `get_methods` возвращает список данных о каждом методе, каждое из которых в свою очередь тоже является списком из трёх элементов: название символической ссылки на метод (для запуска через cl-core), название груп метода (для запуска с помощью клиентов) и отображаемое название метода.

Например:

```
[  
  [cl-install, install, Установка системы],  
  [cl-core-del-request, core_del_request, Удаление запроса],  
  ...  
]
```

- `get_server_host_name` - получение имени хоста, на котором находится сервер утилит.

Не имеет входных параметров, возвращает имя хоста (hostname) сервера утилит, записанное в его сертификате.

- `active_client` - метод используется для графических клиентов, позволяет определить наличие постоянного соединения с каждой из сторон, а так же был ли перезагружен сервер (при этом необходимо переустановить соединение).

Имеет один входной параметр `sid` - номер сессии.

Возвращает:

- значение 0, если обновление информации о сессии прошло успешно.
- значение 1 в случае ошибки;
- значение 2, если клиентом послан неверный идентификатор сессии.

5. Настройка сервера

1. [Перенос учётных записей пользователей в Calculate Directory Server](#)
2. [Настройка LDAP сервера](#)
3. [Использование LDAP сервера для хранения учетных записей](#)
4. [Настройка Samba сервера](#)
5. [Настройка прав доступа ACL](#)
6. [Настройка FTP сервера](#)
7. [Настройка Jabber сервера](#)
8. [Настройка почтового сервера](#)
9. [Настройка Proxy сервера](#)
10. [Настройка DNS сервера](#)
11. [Настройка DHCP сервера](#)
12. [Настройка PXE](#)
13. [Настройка репликации Samba серверов](#)
14. [Настройка репликации почтовых серверов](#)
15. [Настройка сервера шлюза](#)
16. [Настройка Asterisk сервера](#)
17. [Обзор структуры LDAP сервера](#)
18. [Управление клиентскими машинами](#)
19. [Система виртуализации QEMU](#)
20. [Настройка gitolite](#)
21. [Резервное копирование](#)

Перенос учётных записей пользователей в Calculate Directory Server

Рассмотрим более подробно все шаги по переносу пользователей в [Calculate Directory Server](#).

Настройка серверов

Перед настройкой обновите пакет calculate-server до последней версии:

```
emerge calculate-server
```

Настройка сервисов

Перед настройкой сервисов проверьте, что у вас поднята сеть. Для настройки проводной сети воспользуйтесь следующим [руководством](#).

В приведённом примере мы настроим и запустим все поддерживаемые сервисы:

```
cl-setup ldap  
cl-setup unix  
cl-setup samba  
cl-setup mail  
cl-setup jabber
```

Обратите внимание: при настройке вы можете указывать дополнительные параметры. Перед запуском вызовите справку по каждой команде:

```
cl-setup --help  
cl-setup --help-jabber
```

Зададим пароль для входа в домен Linux- и Windows-клиентов

```
cl-passwd --smb client samba  
cl-passwd --smb admin samba
```

Настроим использование distfiles клиентами домена

Создайте каталог /var/calculate/remote/distfiles если его нет

```
mkdir -p /var/calculate/remote/distfiles
```

затем выполните команды

```
cl-groupadd --gid 250 -f portage unix  
cl-usermod -a portage client unix  
chmod 0775 /var/calculate/remote  
chmod -R 2775 /var/calculate/remote/distfiles  
chown -R root:portage /var/calculate/remote/distfiles
```

Настройка Samba

Создадим базовые системные группы

Базовые системные группы могут понадобиться в том случае, если вы захотите ограничить доступ пользователей к определенным ресурсам компьютера. В этом случае системные Unix-группы должны быть продублированы в LDAP-сервере. При необходимости эту операцию можно выполнить позднее.

```
cl-groupadd -f -g 7 lp unix  
cl-groupadd -f -g 10 wheel unix  
cl-groupadd -f -g 18 audio unix  
cl-groupadd -f -g 27 video unix  
cl-groupadd -f -g 35 games unix  
cl-groupadd -f -g 440 plugdev unix  
cl-groupadd -f -g 441 scanner unix  
cl-groupadd -f -g 442 vboxusers unix
```

Более подробно о назначение каждой группы вы можете прочитать [здесь](#).

Добавим группы пользователей

Как минимум у пользователей должна быть одна группа. Тем не менее, если вы впоследствии захотите разграничить доступ к данным пользователям различных отделов, группы - наиболее удобный способ организовать распределенный доступ.

В качестве примера взяты 5 групп:

```
cl-groupadd boss samba  
cl-groupadd job samba  
cl-groupadd it samba  
cl-groupadd logist samba  
cl-groupadd manager samba
```

Настройка Jabber

Вы можете организовать свой Jabber-сервер. В свете недавних событий с ограничением использования ICQ сторонними клиентами это неплохой способ избавиться от зависимости от одной компании. Jabber-сервер может с успехом использоваться и внутри вашей локальной сети для эффективного взаимодействия сотрудников. С целью предотвращения злоупотреблений рабочим временем на сервере предусмотрено логирование сообщений.

Добавим группы пользователей

Создав группы, пользователи автоматически будут размещены в них в вашем Jabber клиенте, что, безусловно, облегчит использование, а также будет служить отправной точкой знакомства со структурой компанией для новых сотрудников.

```
cl-groupadd "Директор" jabber
cl-groupadd "Кадры" jabber
cl-groupadd "Программист" jabber
cl-groupadd "Логист" jabber
cl-groupadd "Менеджер" jabber
```

Добавим пользователей

Ниже приведен простой *bash* скрипт для добавления пользователей. Отредактируйте его, закомментировав неиспользуемые сервисы. Обратите внимание, что используемые в скрипте группы *manager*, *audio*, *lp*, *plugdev*, *scanner* и *video* должны быть предварительно созданы как описано выше.

```
#!/bin/bash

login="ipetrov"
fullname="Петров Иван"
group="manager"
groupJabber="Менеджер"
mail="ip@calculate-linux.org, ip@mail.office.calculate-linux.org"

echo ">>>Настроим Samba"
/usr/bin/cl-useradd -p -c "$fullname" -g $group -G
audio,lp,plugdev,scanner,video $login samba

echo ">>>Настроим Mail"
/usr/bin/cl-useradd -c "$fullname" -p -e $mail $login mail

echo ">>>Настроим Jabber"
/usr/bin/cl-useradd -c "$fullname" -p $login jabber
/usr/bin/cl-usermod -g "$groupJabber" $login jabber
```

В приведенном примере мы создаем два почтовых адреса, т.к. сервер находится в локальной сети и не является основным почтовым сервером.

Настройка LDAP сервера

Введение

LDAP (англ. Lightweight Directory Access Protocol --- «облегчённый протокол доступа к каталогам») --- это сетевой протокол для доступа к службе каталогов. LDAP --- относительно простой протокол, использующий TCP/IP и позволяющий производить операции аутентификации (*bind*), поиска (*search*) и сравнения (*compare*), а также операции добавления, изменения или удаления записей. Обычно LDAP-сервер принимает входящие соединения на порт 389 по протоколам TCP или UDP. Для LDAP-сеансов, инкапсулированных в SSL, обычно используется порт 636.

Настройка

В качестве LDAP сервера в [Calculate Directory Server](#) используется [OpenLDAP](#). Настройка сервера выполняется пакетом *calculate-server* из состава утилит [Calculate 2](#).

Для настройки сервера, воспользуйтесь программой *cl-setup*, выполнив:

```
cl-setup ldap
```

Во время выполнения этой команды, программа выполнит настройку сервера LDAP, запустит его и пропишет в автозагрузку. Будьте внимательны, программа перезапишет базу данных LDAP. Если вы работали с LDAP сервером до этого, выполните [резервное копирование](#) ваших данных. Пароли доступа к LDAP всех сервисов хранятся в `/etc/calculate/calculate.ldap`.

Использование LDAP сервера для хранения учетных записей

Введение

В Unix системе вся информация о пользователях хранится в файле `/etc/passwd`. Это текстовый файл, право на чтение которого имеют все пользователи, а на запись только суперпользователь. Пароли пользователей хранятся в файле `/etc/shadow` в зашифрованном виде, закрытом для чтения и записи. Информация о группах хранится в файле `/etc/group`.

У этого способа хранения есть ряд ограничений, из которых можно выделить сложность переноса пользователей, ограниченность в атрибутах, производительность и т.п. Однако при помощи *PAM* (*Pluggable Authentication Module*), вы можете с легкостью интегрировать в системы UNIX различные технологии аутентификации, в т.ч. *LDAP* (*Lightweight directory Access Protocol*).

Настройка Unix сервера

Выполнить настройки сервера вы можете воспользовавшись программой *cl-setup*, входящей в пакет *calculate-server*. Для этого выполните с правами суперпользователя:

```
cl-setup unix
```

После выполнения этой операции, имена учетных записей пользователей, хранящиеся в базе *LDAP* сервера и имеющие системные *ID*, будут видны в системе.

Добавление и удаление пользователей

Обратите внимание, что после настройки LDAP сервера в качестве хранилища учётных записей Unix пользователей, следует использовать альтернативные команды по управлению пользователями.

Вместо команд: *useradd* (*adduser*), *userdel*, *usermod*, *passwd*, *groupadd*, *groupdel*, *groupmod*, следует использовать альтернативные: *cl-useradd*, *cl-userdel*, *cl-usermod*, *cl-passwd*, *cl-groupadd*, *cl-groupdel*, *cl-groupmod*. Синтаксис этих команд будет во многом совпадать с оригинальными.

Пример добавления пользователя *test*:

```
cl-useradd test unix
```

Пример изменения пароля пользователя *test*:

```
cl-passwd test unix
```

Обратите внимание на опцию *unix*, добавляемую в конце команды.

Настройка Samba сервера

Введение

Samba это популярный пакет программ с открытыми исходными текстами, которая предоставляет файловые и принт-сервисы Microsoft(r) Windows(r) клиентам. При помощи пакета утилит [calculate-server](#) на samba сервере создаются пользователи, группы, ресурсы и происходит управление ими. В качестве Linux клиента может выступать [Calculate Linux Desktop](#) или любой Gentoo дистрибутив с установленным пакетом *calculate-client*. В качестве Windows клиентов могут быть использованы операционные системы семейства Windows. Чтобы настроить сервер и клиент, воспользуйтесь статьей "[переход на использование Linux](#)". Для указания настраиваемого пакета программ в *calculate-server* используется термин - сервис.

Сервис *samba* используется для настройки *Samba*.

Сервис обязательно указывается для программ пакета *calculate-server*.

Пример добавления пользователя *test*:

```
c1-useradd test samba
```

Для пользователя сервера права на файловые ресурсы одинаковы вне зависимости от клиента - Linux или Windows.

Пакет Samba включен в поставку [Calculate Directory Server](#). Если вы используете другую Gentoo систему, пакет можно установить из портежей, выполнив *emerge net-fs/samba*.

Настройка сервера

Настройка сервера выполняется при помощи утилиты *calculate-server*. Для начала убедитесь что у вас [настроен LDAP сервер](#) и выполнены [настройки Unix сервера](#).

Для настройки Samba-сервера выполните команду:

```
c1-setup [параметры] samba
```

В качестве параметров вы можете указать *netbios* и *workgroup*.

- "-n name" - устанавливает имя NetBIOS, под которым будет работать Samba сервер. По умолчанию оно устанавливается равным первому компоненту DNS имени хоста.
- "-w workgroup" - имя домена или рабочей группы NT для компьютеров, которые будут получать доступ к этому серверу.

Если Samba сервер будет выступать в качестве *PDC (первичного контроллера Windows домена)*, вам следует задать пароль администратора сервера - пользователя с логином *admin*.

```
c1-passwd --smb admin samba
```

Пользователь *admin* используется только для ввода клиентского windows компьютера в домен. Он не имеет домашней директории.

Если нужен администратор домена для управления windows компьютерами, добавьте нового пользователя который будет включен в доменную группу "Domain Admins", или включите в эту группу существующего пользователя.

Пример создания администратора домена:

```
c1-useradd -p --gid "Domain Admins" -c "Администратор домена" d_admin  
samba
```

Для [подключения unix клиентов](#), укажите пароль для служебного пользователя *client*.

```
c1-passwd --smb client samba
```

Добавление и удаление пользователей

Для работы с пользователями, используйте аналоги стандартных Unix команд: *cl-useradd*, *cl-userdel*, *cl-usermod*, *cl-passwd*, *cl-groupadd*, *cl-groupdel*, *cl-groupmod*. Синтаксис этих команд будет во многом совпадать с оригинальными.

Вместо *smbpasswd*, используйте *cl-passwd* для изменения паролей пользователей, в т.ч. администратора Windows компьютеров.

Пример добавления пользователя test:

```
cl-useradd test samba
```

Пример изменения пароля пользователя test:

```
cl-passwd test samba
```

Пример добавления пользователя с первичной группой Domain Admins - администраторы домена

```
cl-useradd -g 'Domain Admins' test samba
```

Обратите внимание на опцию *samba*, добавляемую в конце команды.

Настройка прав доступа

Настройка прав доступа к файловой системе

Для настройки прав доступа к файлам на сервере используйте ACL (Access Control List --- список контроля доступа). Изменяя права на файлы, вы ограничиваете к ним доступ в равной степени как для Windows, так и для Linux клиентов.

Права доступа применяются как к файлам, так и к директориям. Вы можете указать права доступа на владельца файла, либо на группу. Если Windows клиент будет распознавать только Samba группы, то в Linux будут отображаться имена Unix и Samba группы. Поэтому, для разграничения прав доступа, предпочтительней использовать Samba группы.

Для создания Samba группы "manager", выполните:

```
cl-groupadd manager samba
```

для создания Unix группы "job", выполните:

```
cl-groupadd job unix
```

Подробно настройка прав доступа с использованием ACL описана в статье ["Настройка прав доступа файловой системы"](#).

Настройка прав доступа для пользователей Windows компьютеров

Настройка прав доступа к общим файлам сервера описана выше.

Для настройки дополнительных прав Windows машин, таких как: возможность устанавливать программы, возможность выхода из домена и т.д., используйте Samba группы.

Пример повышения прав пользователя test до администратора домена:

```
cl-groupmod -a test 'Domain Admins' samba
```

Структура Samba групп

Samba группы могут быть следующих типов:

- Доменные группы (номер типа группы 2)
- Локальные группы (номер типа группы 4)
- Встроенные группы (номер типа группы 5)

Группы созданные по умолчанию

Доменные группы Доменные группы - глобальные группы которые действуют в домене.

- *Domain Admins* - администраторы домена (полные права на компьютерах в домене).
- *Domain Guests* - гости домена (минимальные права).
- *Domain Users* - пользователи домена.
- *Domain Computers* - компьютеры домена.

Локальные группы

Локальные группы - группы действующие локально на данном компьютере.

Локальные группы отсутствуют.

Встроенные группы

Встроенные группы - группы встроенные в систему.

- *Administrators* - администраторы (полные права)
- *Account Operators* - операторы учетных записей. Создание и управление пользовательской учетной информацией, создание и управление группами, резервное копирование файлов и каталогов.
- *Backup Operators* - операторы архивов. Резервное копирование, восстановление из резервной копии, остановка системы.
- *Print Operators* - операторы печати. Управление принтерами, резервное копирование.
- *Replicators* - репликаторы. Эта группа используется службой репликации File Replication на контроллерах домена.
- *System Operators* - операторы системы. Изменение системного времени, останов системы, останов с удаленной системы, резервное копирование, восстановление из резервной копии, блокирование сервера, преодоление блокировки сервера, форматирование жесткого диска, управление сетевыми каталогами, управление принтерами.

Создание Samba группы

Создание доменной группы test. По умолчанию создается доменная группа, тип группы - 2.

```
cl-groupadd test samba
```

Создание встроенной группы 'Power Users' - пользователи имеющие дополнительные права.

```
cl-groupadd -g 547 --rid 547 -t 5 'Power Users' samba
```

Где:

- *g* - идентификатор группы 547 (Group ID)
- *rid* - уникальный идентификатор 547 (RID)
- *t* - тип группы 5 (встроенная группа)

Просмотр информации

Для просмотра информации о пользователях и группах сервера воспользуемся командой *cl-info*:

Перечень пользователей unix сервиса:

```
cl-info -u unix
```

Перечень пользователей samba сервиса:

```
cl-info -u samba
```

Просмотр информации о пользователе unix сервиса:

```
cl-info -U <название пользователя> unix
```

Просмотр информации о пользователе samba сервиса:

```
cl-info -U <название пользователя> samba
```

Перечень существующих групп unix сервиса:

```
cl-info -g unix
```

Перечень существующих групп samba сервиса:

```
cl-info -g samba
```

Просмотр информации о группе unix сервиса:

```
cl-info -G <название группы> unix
```

Просмотр информации о группе samba сервиса:

```
cl-info -G <название группы> samba
```

Настройка прав доступа ACL

Для создания прав доступа к файлам Samba ресурса сервера используйте *ACL* (*Access Control List* --- список контроля доступа). При помощи ACL вы можете установить доступ к файлам определенным группам, либо конечным пользователям.

Стандартные команды работы с ACL - *setfacl* и *getfacl* - подробно описаны в руководстве, поэтому здесь мы ограничимся примерами.

После [настройки Samba сервера](#), создаётся путь для общей точки монтирования */var/calculate/server-data/samba/share*.

Внимание. Убедитесь, что файловая система диска, содержащего указанную директорию, поддерживает ACL. Не забудьте в опции монтирования файловой системы указать параметр "acl" (для xfs не требуется).

Новый ресурс

Для примера допустим у вас в системе есть две группы *manager* и *logist*, а также два пользователя *iivanov* и *apetrov*.

Создадим директорию Менеджер для доступа на чтение и запись пользователям, входящим в группу *manager*.

```
mkdir -m 700 Менеджер  
setfacl -m d:g:manager:rwx,g:manager:rwx Менеджер
```

Первая команда создаст директорию с правами чтения выполнения и изменения содержимого для пользователя root. Второй командой мы установим те же права для пользователей группы *manager*.

Файлы и директории, создаваемые пользователями этой группы, либо root-ом, будут наследовать атрибуты доступа.

Просмотр прав доступа

Права на доступ к директории, вы можете посмотреть из консоли, либо через файловый менеджер dolphin, с [Calculate Linux Desktop](#) (KDE версия).

Увидеть права доступа из консоли можно выполнив команду:

```
getfacl Менеджер
```

Программа выведет следующий текст:

```
# file: \320\234\320\265\320\275\320\265\320\264\320\266\320\265\321\200/
# owner: root
# group: root
user::rwx
group::---
group:manager:rwx
mask::rwx
other::---
default:user::rwx
default:group::---
default:group:manager:rwx
default:mask::rwx
default:other::---
```

Для просмотра прав из dolphin, нажмите на файле или директории правой кнопкой мыши, выберите вкладку Права, далее нажмите на кнопку *Дополнительные права*. Растворите открывшееся окно, чтобы увидеть все атрибуты.

Управление доступом

Для управлением доступом используется команда *setfacl*.

Для модификации или добавления правила используется параметр -m.

```
-m user:[пользователь]:права[,user:пользователь:права]
-m group:[группа]:права[,group:группа:права]
```

Если *пользователь* пропущен, то *права* назначаются владельцу файла.

Если *группа* пропущена, то *права* назначаются группе-владельцу файла.

Пример

Добавить право на чтение/запись файла *secretinfo* пользователям *iivanov* и *apetrov*:

```
setfacl -m user:iivanov:rw,u:apetrov:rw secretinfo
```

Добавить право на чтение/выполнение файла *runit* группе

Настройка FTP сервера

FTP (англ. File Transfer Protocol --- протокол передачи файлов) --- протокол, предназначенный для передачи файлов в компьютерных сетях. FTP позволяет подключаться к серверам FTP, просматривать содержимое каталогов и загружать файлы с сервера или на сервер.

Настройка FTP сервера производится в несколько этапов:

Установка FTP сервиса в систему

Настройка сервиса *ftp* требует, чтобы в системе были установлены сервисы *ldap* и *unix*, поэтому если они еще не были установлены устанавливаем их командами:

```
cl-setup ldap
cl-setup unix
```

Установка *ftp* сервиса производится командой:

```
cl-setup ftp
```

После чего *ftp* становится доступен для пользователя *anonymous*. На сервере также автоматически в *ftp* директории создаются папки *tmp* и *pub*.

Для подключения к ftp откройте любой браузер, поддерживающий этот протокол, и введите в адрес страницы `ftp:// вашСервер.`

Учетные записи

Учетные записи используются для авторизованного доступа к FTP серверу. Для каждой из них может быть настроена домашняя папка, при использовании совместно с unix сервисом, разграничены права.

Управление учетными записями

Добавление учетной записи

Для добавления учетной записи сервиса ftp используется команда `cl-useradd:`

- Создать учетную запись user1 и задать пароль

```
cl-useradd -p user1 ftp
```

- Создать учетную запись user1 с домашней директорией, задать пароль. Домашняя директория по умолчанию помещается в `pub/users/<имя_учетной_записи>` домашней директории сервиса ftp. К этой директории имеет доступ только этот пользователь.

```
cl-useradd -p -m user1 ftp
```

- Создать учетную запись user1 с указанной домашней директорией, задать пароль. Домашняя директория в этом случае будет находиться по указанному пути относительно корня домашней директории сервиса ftp.

```
cl-useradd -p -m -d pub/user1 user1 ftp
```

Смена пароля учетной записи

Смена пароля учетной записи сервиса ftp производится командой `cl-passwd:`

```
cl-passwd user1 ftp
```

Удаление учетной записи

Удаление учетной записи из сервиса ftp производится командой `cl-userdel`, при этом удаляется пользовательская домашняя директория.

```
cl-userdel user1 ftp
```

P.S. Т.к. при создании учетной записи сервиса ftp, автоматически создается учетная запись сервиса unix, то при удалении учетной записи сервиса ftp, учетная запись unix остается.

Управление правами доступа к директориям

Права доступа - это атрибуты файла или директории, которые указывают серверу, кто и что может делать с соответствующим файлом или директорией.

Файлы имеют двух владельцев: пользователя (*user owner*) и группу пользователей (*group owner*). Для каждого файла есть индивидуальные права доступа, которые разбиты на три группы: Доступ для пользователя-владельца файла (*owner*). Доступ для группы-владельца файла (*group*). Доступ для остальных пользователей (*others*).

Для каждой категории устанавливаются три вида доступа: (x) - право на запуск файла/право входа в директорию, (r) - право на чтение файла/директории, (w) - право на изменение (редактирование) файла, удаление/создание файлов в директории.

Для определения прав доступа к директориям сервис ftp взаимодействует с unix сервисом: каждой учетной записи ftp соответствует одноименная учетная запись unix. Таким образом, пользователь, входя под своей учетной записей на ftp сервер, получает определенный доступ к файлу в зависимости от того, является ли он пользователем-владельцем файла, и входит ли в группу-владельца файла. Если пользователь не является владельцем и не входит в группу-владельца, то доступ определяется по правам *others*. В случае анонимного доступа к ftp, права также определяются по *others*.

Установка прав доступа

Установка прав осуществляется на сервере командной chmod

```
# установить права user=все group=чтение/запись others=чтение
chmod u=rwx, g=rw, o=r file
```

```
# установить группу-владельца pubwriter
chgrp pubwriter pub
# установить разрешение писать в директорию pub для группы
chmod g+w pub
```

Управление unix группами

Так как ftp сервис тесно связан с unix сервисом, то управление группами, в которые входит пользователь осуществляется через unix сервис.

Добавление группы

Добавление группы осуществляется командой *cl-groupadd*:

```
# добавить группу pubwriter
cl-groupadd pubwriter unix
```

Удаление группы

Удаление группы осуществляется командой *cl-groupdel*:

```
# удалить группу test
cl-groupdel test unix
```

Добавление/удаление учетных записей в группу

Добавление и удаление учетных записей производится двумя способами:

- с использованием команды *cl-usermod*:

```
# добавить пользователя guest в группы pubwriter
cl-usermod -a pubwriter guest unix
```

```
# заменить пользователю guest группы на guest
cl-usermod -G guest guest unix
```

- с использованием команды *cl-groupmod*:

```
# удалить пользователей guest1 и guest2 из группы pubwriter
cl-groupmod -d guest1,guest2 pubwriter unix
```

```
# добавить пользователей guest1,guest2 в группу pubwriter
cl-groupmod -a guest1,guest2 pubwriter unix
```

Настройка Jabber сервера

XMPP (ранее известный как Jabber) --- Extensible Messaging and Presence Protocol (англ. расширяемый протокол обмена сообщениями и информацией о присутствии), это основанный на XML открытый, свободный для использования протокол для мгновенного обмена сообщениями и информацией о присутствии в режиме, близкому к режиму реального времени.

Настройка jabber сервера производится в несколько этапов:

Установка Jabber сервиса в систему

Jabber сервис, настраиваемый с помощью пакета Calculate-Server, требует, чтобы в системе был установлен сервис ldap. Если сервис не был ранее установлен, установим его командой:

```
cl-setup ldap
```

Установка jabber сервиса производится командой

```
cl-setup jabber
```

В этом случае сервис будет установлен с параметрами по умолчанию: имя хоста jabber сервиса - имя хоста машины, лог сообщений пользователей вестись не будет.

Для указания дополнительных jabber хостов (например `jabber.myhost.ru`) используется параметр `--hosts`:

```
cl-setup --hosts jabber.myhost.ru jabber
```

Установка сервиса с ведением лога сообщений пользователей, производится командой:

```
cl-setup --history on jabber
```

Сообщения будут сохраняться в директорию `/var/log/jabber`

При установке сервиса потребуется пароль для учетной записи `admin`.

Учетные записи

Каждый пользователь в сети имеет уникальный идентификатор --- *Jabber ID* (сокращенно JID). Адрес JID, подобно адресу электронной почты, содержит имя пользователя и доменное имя сервера, на котором зарегистрирован пользователь, разделённые знаком @. Например, пользователь user, зарегистрированный на сервере example.com, будет иметь адрес: user@example.com.

Управление учетными записями

Добавление учетной записи

Добавление учетной записи сервиса jabber осуществляется командой `cl-useradd`

```
# добавить пользователя guest@домен с Nickname Гость  
cl-useradd -p -c "Гость" guest jabber  
# добавить пользователя guest@домен с Nickname Гость, и установить для  
него фотографию  
cl-useradd -p -c "Гость" -i pic/guest.png guest jabber
```

Поддерживаемые форматы изображений определяются возможностями *ImageMagick*. Если в системе его нет - доступен только формат jpg.

Смена пароля

Смена пароля учетной записи сервиса jabber осуществляется командой `cl-passwd`

```
cl-passwd guest jabber
```

Блокировка и удаление учетной записи

Удаление учетной записи из сервиса jabber осуществляется командой *cl-userdel*
cl-userdel guest jabber

Блокировка учетной записи производится командой *cl-usermod -L*
cl-usermod -L guest jabber

Разблокировать учетную запись можно командой *cl-usermod -U*
cl-usermod -U guest jabber

Группы

Группа - набор ID, использующийся для рассылок сообщений нескольким пользователям одновременно. При подключении пользователя к jabber сервису он автоматически получит список групп и их участников. Пользователей без группы в список контактов добавлять придется вручную. Пользователь может входить только в одну группу.

Управление группами

Создание группы

Создание группы сервиса jabber осуществляется командой *cl-groupadd*
создать группу с названием "Тестовая группа"
cl-groupadd "Тестовая группа" jabber

Имя группы будет отображаться у пользователей в списке контактов.

Удаление группы

Удалить группу из сервиса jabber можно командой *cl-groupdel*
удалить "Тестовую группу"
cl-groupdel "Тестовая группа" jabber

Удаляется только группа, все пользователи, которые были включены в эту группу остаются без группы.

Переименование группы

Переименовать существующую группу сервиса jabber можно командой *cl-groupmod*
назначить группе 'My test' новое имя 'Тестовая группа'
cl-groupmod -n 'Тестовая группа' 'My test' jabber

Изменение состава группы

Изменять состав jabber группы можно при помощи команд *cl-groupmod* и *cl-usermod*
Поместить пользователя guest в группу 'Guest group'
cl-usermod -g "Guest group" guest jabber
Удалить пользователя guest из группы 'Guest group'
cl-groupmod -d guest "Guest group" jabber
Добавить пользователей guest, guest2 в группу 'Guest group'
cl-groupmod -a guest, guest2 "Guest group" jabber

Ограничения

Если на вашем сервере больше одного сетевого интерфейса, и вы желаете чтобы он работал только на одном из них, внесите в его конфиг соответствующую директиву {ip, {xxx, xxx, xxx, xxx}}, в раздел "listen". Обратите внимание! Разряды IP отделяются не точками, а запятыми!

```
{5223, ejabberd_c2s, [
    {access, c2s},
    {shaper, c2s_shaper},
    {ip, {192, 168, 1, 6}},
    {certfile, "/etc/jabber/ssl.pem"}, tls,
    {max_stanza_size, 65536}
]},
```

Соответственно, эту директиву можно прописать во всех службах ejabberd: ejabberd_s2s, ejabberd_http, и т д.

Настройка почтового сервера

Установка почтового сервиса в систему

Почтовый сервис mail требует, чтобы в системе были установлены сервисы LDAP и Unix, поэтому если они еще не были установлены установите их командой:

```
cl-setup ldap
cl-setup unix
```

Установка почтового сервиса производится командой:

```
# установка сервиса с параметрами по умолчанию
cl-setup mail
```

Для указания имени почтового хоста, отличного от используемого по умолчанию, служит параметр "--host":

```
# mymail.mydomain.com почтовый хост (не указывайте одно короткое имя,
например: mymail)
cl-setup --host mymail.mydomain.com mail
```

Использование протокола pop3 и/или imap управляется через параметр "--type":

```
# устанавливает сервис с поддержкой двух протоколов pop3 и imap с
шифрованием TLS
cl-setup --type imap,pop3 mail
# устанавливает сервис с поддержкой только pop3 и шифрованием TLS
cl-setup --type pop3 mail
```

Шифрование (или отмена его использования) указывается параметром "--crypt":

```
# будет использовано шифрование TLS, и по умолчанию будет использоваться
только IMAP
cl-setup --crypt tls mail
# без шифрования, по умолчанию используется только IMAP
cl-setup --crypt none mail
```

Если необходимо указать используемые протоколы и шифрование, выполним команду с параметрами следующего вида:

```
# без шифрования, поддержка imap и pop3  
cl-setup --crypt none --type imap,pop3 mail
```

После выполнения команды *cl-setup*, с требуемым набором параметров - почтовый сервис конфигурируется и запускается. Результат можно проверить, посмотрев открытые порты:

```
> netstat -tln  
...  
tcp      0      0 *:imaps          *:*                  LISTEN  
tcp      0      0 *:pop3s         *:*                  LISTEN  
tcp      0      0 *:smtp          *:*                  LISTEN  
...  
...
```

Для предотвращения спама для системных почтовых пользователей */etc/mail/aliases* не создается почтовая директория.

Для того чтобы получать письма посланные на адреса указанные в */etc/mail/aliases* выполните следующие команды:

```
mkdir /var/calculate/server-data/mail/nobody  
chown nobody:nobody /var/calculate/server-data/mail/nobody  
chmod 0700 /var/calculate/server-data/mail/nobody
```

Письма для системных почтовых пользователей будут находиться в */var/calculate/server-data/mail/nobody*

Учетные записи

Учётная запись - это запись, которая содержит сведения, необходимые для идентификации пользователя при подключении к системе, информацию для авторизации и учёта. В данном случае для подключения к почтовому сервису или почтовому ящику.

Почтовый ящик - логически выделенная часть дискового пространства, предназначенная для хранения электронных почтовых сообщений, которая обозначается электронным почтовым адресом. Почтовый ящик может иметь несколько почтовых адресов, называемых синонимами почтового адреса (псевдонимами почтового адреса).

Управление учетными записями

Добавление учетных записей

Добавление учетной записи пользователя почтового сервиса производится командой *cl-useradd*:

```
cl-useradd -p -e <почтовый псевдоним один или несколько через запятую>  
<учетная запись> mail
```

```
# добавляем пользователя guest с псевдонимом для почты  
guestmail@mymail.mydomain.com
```

```
cl-useradd -p -e guestmail@mymail.mydomain.com guest mail
```

Смена пароля

Для смены пароля учетной записи почтового сервиса используется команда *cl-passwd*:

```
cl-passwd guest mail
```

Удаление учетных записей

Удаление учетной записи пользователя почтового сервиса производится командой *cl-userdel*

```
cl-userdel guest mail
```

Почтовые группы

Почтовая группа --- набор почтовых адресов, использующийся для рассылок почты нескольким корреспондентам. Письмо, отправленное на адрес группы, рассыпается для всех почтовых учетных записей, входящим в эту группу.

Управление почтовыми группами

Добавление почтовой группы

Добавление почтовой группы производится командой *cl-groupadd*

```
# добавить почтовую группу guestgroup с альтернативным почтовым адресом gg@mydomain.com
```

```
cl-groupadd -e gg@mydomain.com guestgroup mail
```

Удаление почтовой группы

Удаление почтовой группы производится командой *cl-groupdel*:

```
# удалить почтовую группу guestgroup
```

```
cl-groupdel guestgroup mail
```

Добавление и удаление почтовых учетных записей в группу

Добавление и удаление учетных записей производится двумя способами:

- с использованием команды *cl-usermod*

```
# добавить пользователя guest в группы guesttest и guestgroup
```

```
cl-usermod -a guesttest,guestgroup guest mail
```

```
# заменить пользователю guest группы на guesttest
```

```
cl-usermod -G guesttest guest mail
```

- с использованием команды *cl-groupmod*

```
# удалить пользователей guest1 и guest2 из группы guesttest
```

```
cl-groupmod -d guest1,guest2 guesttest mail
```

```
# добавить пользователей guest1 и guest2 в группу guesttest
```

```
cl-groupmod -a guest1,guest2 guesttest mail
```

Проверка работы сервера

Проверить работоспособность сервера можно при помощи программы telnet, при этом сервер должен быть сконфигурирован без шифрования ("--crypt none"). В конечном рабочем варианте, для безопасности, шифрование следует оставить включенным.

1. устанавливаем сервис

```
cl-setup --type imap,pop3 --crypt none mail
```

1. добавляем пользователя guest

```
cl-useradd -p -e guest@mymail.mydomain.org guest mail
```

1. вводим пароль

2. запускаем telnet

```
> telnet
```

1. подключаемся к smtp

```
> open localhost 25
Trying 127.0.0.1...
Connected to mymail.mydomain.org.
Escape character is '^]'.
220 mymail.mydomain.org ESMTP
> EHLO "mymail"
250-mymail.mydomain.org
250-PIPELINING
250-SIZE 100000000
250-ETRN
250-ENHANCEDSTATUSCODES
250-8BITMIME
250 DSN
> MAIL FROM:<noname@mailer.org>
250 2.1.0 Ok
> RCPT TO:<guest@mymail.mydomain.org>
250 2.1.5 Ok
> DATA
354 End data with <CR><LF>.<CR><LF>
> Hello
> .
250 2.0.0 Ok: queued as D42932B91
> QUIT
221 2.0.0 Bye
Connection closed by foreign host.
```

1. теперь запускаем telnet и подключаемся по pop3

```
> telnet
> open localhost 110
Trying 127.0.0.1...
Connected to mymail.mydomain.org.
Escape character is '^]'.
+OK Dovecot ready.
> USER guest
+OK
> PASS 111
+OK "Аутентифицироваться удалось"
> LIST
+OK 1 messages: "На сервере есть одно сообщение"
1 529
.
QUIT
+OK Logging out.
Connection closed by foreign host.
```

Настройка Proxy сервера

Настройка Proxy сервера производится в несколько этапов:

Установка Proxy сервиса в систему

Поддержка proxy сервиса появилась в пакете *Calculate-server* в версии 2.0.13. В качестве сервера используется наиболее распространенный proxy сервер *Squid*.

Перед установкой, убедитесь что у вас в системе установлен сервис LDAP, если он не был установлен, выполните установку командой:

```
cl-setup ldap
```

Установка proxy сервиса производится командой

```
cl-setup proxy
```

В этом случае сервис будет установлен с параметрами по умолчанию: имя хоста proxy сервиса - имя хоста машины, порт для соединения - 8080.

Установка сервиса с установкой доверительных сетей:

```
cl-setup -a proxy
```

При установке сервиса будут созданы базовые группы:

- *http* - доступный порт 80
- *ftp* - доступный порт 21
- *https* - доступный порт 443
- *gopher* - доступный порт 70
- *wais* - доступный порт 210
- *unregistered* - диапазон доступных портов 1025-65535
- *http-mgmt* - доступный порт 280
- *gss-http* - доступный порт 488
- *filemaker* - доступный порт 591
- *multiling* - доступный порт 777
- *swat* - доступный порт 901

Интервал обновления кэша изменений в правах доступа пользователей по умолчанию составляет 5 минут.

Управление учетными записями

Добавление учетной записи

Добавление учетной записи пользователя proxy сервиса осуществляется командой *cl-useradd*

```
# добавим пользователя _ivan_ с полным именем в комментарии
```

```
cl-useradd -p -c "Ivanov Ivan" ivan proxy
```

Смена пароля

Смена пароля учетной записи осуществляется командой *cl-passwd*, пример:

```
cl-passwd ivan proxy
```

Удаление/блокировка учетной записи

Удаление учетной записи осуществляется командой *cl-userdel*, пример

```
cl-userdel ivan proxy
```

Блокировка учетной записи производится командой *cl-usermod -L*, пример

```
cl-usermod -L ivan proxy
```

Разблокировать учетную запись можно командой *cl-usermod -U*, пример
cl-usermod -U ivan proxy

Группы

Группа - набор учетных записей для применения правил доступа.
У группы есть параметр определяющий диапазон сетевых портов.
Пользователю разрешен доступ к ресурсу только в том случае, если доступ к порту, по которому происходит обращение, описывается одной из групп пользователя.

Управление группами

Создание группы

Создание группы proxy сервиса осуществляется командой *cl-groupadd*, пример создания группы "adsl":
cl-groupadd -p 80,83,2000-3000 adsl proxy

Пользователи добавленные в эту группу будут иметь доступ к сетевому порту 80 и 83, а также к диапазону сетевых портов 2000-3000, по которым может работать определенных сервис. Условно мы его называли "adsl".

Удаление группы

Удалить группу из сервиса можно командой *cl-groupdel*, пример:
cl-groupdel adsl proxy

Переименование группы

Переименовать существующую группу сервиса можно командой *cl-groupmod*, пример смены имени группы "adsl" на "adsl2":

cl-groupmod -n adsl2 adsl proxy

Изменение состава группы

Изменить описания групп можно при помощи команд "cl-groupmod" и "cl-usermod". Примеры:
Добавим пользователя ivan в группу 'http' (откроем пользователю доступ к сайтам)
cl-usermod -G http ivan proxy

Удалим пользователя ivan из группы 'http' (закроем доступ к сайтам)
cl-groupmod -d ivan http proxy

Добавим пользователей ivan и guest в группу http
cl-groupmod -a ivan,guest http proxy

Для того чтобы дать пользователю доступ к ftp, он должен иметь доступ к порту 21 и порту proxy сервера (по умолчанию 8080).

Дать пользователю guest доступ к ftp
cl-usermod -G ftp,unregistered guest proxy

Настройка DNS сервера

- [Настройка DNS сервера](#)

- [Установка DNS сервиса в систему](#)
- [Управление DNS сервисом](#)
- [Создание DNS зоны](#)
- [Модификация DNS зоны](#)
- [Удаление DNS зоны](#)
- [Удаление MX записей для зоны](#)
- [Удаление A записи для зоны](#)
- [Создание DNS записи](#)
- [Модификация записи](#)
- [Удаление DNS записи](#)
- [Информация о DNS сервисе](#)
- [Информация о всех зонах](#)
- [Информация о зоне](#)
- [Информация о записи](#)
- [Пример создания зоны и записей в ней](#)
- [Проверка работоспособности DNS сервера](#)

Настройка DNS сервера производится в несколько этапов:

Установка DNS сервиса в систему

Поддержка DNS сервиса появилась в пакете [calculate-server 2.1.4](#).

В качестве сервера используется наиболее распространенный DNS сервер BIND.

Перед установкой убедитесь что BIND скомпилирован с поддержкой *sdb-ldap*.

Перед установкой, убедитесь что у вас в системе установлен сервис ldap, если он не был установлен, выполните установку командой:

```
cl-setup ldap
```

Установка dns сервиса выполняется командой

```
cl-setup dns
```

Установка сервиса с установкой доверительных сетей:

```
cl-setup -a dns
```

Примечание: Интервал времени жизни DNS записи - ttl, составляет 178600 секунд.

Управление DNS сервисом

Термины:

- *DNS зона* - сегмент пространства доменных имен.
- *master DNS зона* - основная зона хранения записей.
- *slave DNS зона* - подчиненная основной зона хранения записей.
- *Прямая DNS зона* - зона хранения записей соответствия доменного имени ip адресу.
- *Обратная DNS зона* - зона хранения записей соответствия ip адреса доменному имени.
- *Авторитетивный сервер* - сервер для хранения DNS зоны, записи которого считаются авторитетными для других DNS серверов.
- *SOA запись* - запись описания зоны.
- *NS запись* - доменное имя авторитетивного сервера.
- *A запись* - соответствие доменного имени ip адресу.
- *PTR запись* - соответствие ip адреса доменному имени.
- *CNAME запись* - соответствие одного доменного имени другому.
- *MX запись* - соответствие доменного имени доменным именам почтовых серверов.

Создание DNS зоны

Для создания DNS зоны используется команда `cl-dns-zoneadd`.

Создание master DNS зоны

Создание зоны с авторитативным сервером в создаваемой зоне.

```
cl-dns-zoneadd -n <имя зоны> --server <имя авторитативного сервера>
--ipserver <ip авторитативного сервера>
```

Создание зоны с авторитативным сервером в другой зоне.

```
cl-dns-zoneadd -n <имя зоны> --server <имя авторитативного сервера>
```

Примеры:

```
cl-dns-zoneadd -n test.ru --server test.ru --ipserver 10.0.0.34
```

- Будет создана прямая зона test.ru
- Если не существует, будет создана обратная зона 0.0.10.in-addr.arpa
- Будет создана в записи зоны test.ru, A запись - test.ru соответствует 10.0.0.34
- Будет создана в записи зоны test.ru NS запись - test.ru
- Если обратная зона 0.0.10.in-addr.arpa была создана, для нее будет создана NS запись - test.ru

```
cl-dns-zoneadd -n test.ru --server ns.test.ru --ipserver 10.0.0.34
```

- Будет создана прямая зона test.ru
- Если не существует, будет создана обратная зона 0.0.10.in-addr.arpa
- Будет создана в зоне test.ru, A запись - ns.test.ru соответствует 10.0.0.34
- Будет создана в записи зоны test.ru NS запись - ns.test.ru
- Если обратная зона 0.0.10.in-addr.arpa была создана, для нее будет создана NS запись - test.ru
- В обратной зоне 0.0.10.in-addr.arpa будет создана PTR запись - 10.0.0.34 соответствует ns.test.ru, если такая запись не существует.

```
cl-dns-zoneadd -n 10.0.10.0/24 --server test.ru
```

* Будет создана обратная зона для сети 10.0.10.0/24 - 10.0.10.in-addr.arpa * Будет создана в записи зоны 10.0.10.in-addr.arpa NS запись - test.ru

Создание slave DNS зоны

Создание DNS зоны.

```
cl-dns-zoneadd -t slave -n <имя зоны> --servers <ip серверов хранения
master зоны для этой зоны>
```

Примеры:

```
cl-dns-zoneadd -t slave -n slave.ru --servers 10.0.0.3,10.0.10.5
```

- Будет создана подчиненная прямая зона slave.ru, данные для которой будут получены из основной зоны slave.ru находящейся на DNS серверах с адресами 10.0.0.3, 10.0.10.5

```
cl-dns-zoneadd -t slave -n 10.0.0.0/24 --servers 10.0.0.3
```

* Будет создана подчиненная обратная зона для сети 10.0.0.0/24 - 0.0.10.in-addr.arpa, данные для которой будут получены из основной зоны 0.0.10.in-addr.arpa находящейся на DNS сервере с адресом 10.0.0.3

Модификация DNS зоны

Для модификации DNS зоны используется команда *cl-dns-zonemod*.

Модификация параметров зоны возможна только для master зоны.

```
cl-dns-zonemod -n <имя зоны или сеть> <параметры>
```

имя зоны - модификация прямой зоны

сеть - модификация обратной зоны

Параметры модификации зоны:

- --server - изменение доменного имени главного авторитативного сервера зоны
- --ip - изменение или добавление в случае отсутствия, ip адреса для зоны (модификация или добавление A записи)
- --mx - замена или добавление в случае отсутствия, MX записей для зоны (модификация или добавление доменных имен почтовых серверов)
- --mxmod - замена одного доменного имени почтового сервера на другой в MX записи для зоны (модификация доменного имени почтового сервера)
- --email - изменение почтового адреса администратора зоны (по умолчанию root@имя_зоны)
- --servers - изменение списка всех авторитативных серверов зоны (NS записи зоны)
- --refresh - интервал времени после которого будет обновлена зона в секундах или число + (M - минуты, H - часы, D - дни, W - недели).
По умолчанию 8H - 8 часов.
- --update - интервал времени после неудачного обновления зоны после которого будет сделано новое обновление зоны.
По умолчанию 2H - 2 часа.
- --expiry - интервал времени после которого данные зоны устареют на вторичных DNS серверах в случае невозможности соединения с главным DNS сервером.
По умолчанию 2W - 2 недели.
- --minimum - интервал времени хранения данных неудачных запросов для этой зоны.
По умолчанию 2H - 2 часа.

Примеры:

```
cl-dns-zonemod -n test.ru --email admin@test.ru
```

Модификация почтового адреса администратора зоны

```
cl-dns-zonemod -n test.ru --refresh 10H
```

Модификация интервала времени обновления зоны (10 часов)

Удаление DNS зоны

Для удаления DNS зоны используется команда *cl-dns-zonedel*.

```
cl-dns-zonedel -n <имя зоны или сеть>
```

имя зоны - удаление прямой зоны

сеть - удаление обратной зоны

Примеры:

```
cl-dns-zonedel -n test.ru
```

Будет удалена прямая зона test.ru

```
cl-dns-zonedel -n 10.0.0.0/24
```

Будет удалена обратная зона 0.0.10.in-addr.arpa

Удаление MX записей для зоны

Пример.

```
cl-dns-zonedel --mx -n test.ru
```

Будут удалены MX записи для зоны test.ru (доменные имена почтовых серверов для зоны)

Удаление A записи для зоны

Пример.

```
cl-dns-zonedel --ip -n test.ru
```

Будет удалена A запись для зоны test.ru (ip зоны)

Создание DNS записи

Для создания DNS записи используется команда *cl-dns-recadd*.

Для создания записи необходимо создание master зоны в которую будет добавлена эта запись.

Для A записи (host.test.ru --> 10.0.0.4) необходимо создание master прямой зоны test.ru.

Для PTR записи (10.0.0.4 --> host.test.ru) необходимо создание master обратной зоны 0.0.10.in-addr.arpa

Создание A записи

Примеры создания записей: Создание A записи и PTR записи. Сначала должны быть созданы прямая и обратная зоны, test.ru и 0.0.10.in-addr.arpa.

```
cl-dns-recadd --host host.test.ru --ip 10.0.0.66
```

- Будет создана запись в прямой зоне test.ru, host.test.ru соответствует 10.0.0.66.
- Будет создана запись в обратной зоне 0.0.10.in-addr.arpa, 10.0.0.66 соответствует host.test.ru

Создание только A записи. Сначала должна быть создана прямая зона test.ru.

```
cl-dns-recadd --autoptr off --host host.test.ru --ip 10.0.0.66
```

- Будет создана запись в прямой зоне test.ru, host.test.ru соответствует 10.0.0.66.

Создание A записи, MX записи и PTR записи

Пример создания A записи, MX записи и PTR записи. Сначала должны быть созданы прямая и обратная зоны, test.ru и 0.0.10.in-addr.arpa.

```
cl-dns-recadd --mx mail1.test.ru,mail2.test.ru --host host2.test.ru --ip 10.0.0.69
```

- Будет создана запись в прямой зоне test.ru, host2.test.ru соответствует 10.0.0.69.
- Будет создана MX запись в прямой зоне test.ru, host2.test.ru соответствует двум почтовым серверам mail1.test.ru (приоритет 10), mail2.test.ru (приоритет 20)
- Будет создана запись в обратной зоне 0.0.10.in-addr.arpa, 10.0.0.69 соответствует host2.test.ru

Создание A записи и MX записи

Пример создания A записи и MX записи. Сначала должна быть создана прямая зона test.ru.

```
cl-dns-recadd --autoptr off --mx mail1.test.ru,mail2.test.ru --host host2.test.ru --ip 10.0.0.69
```

- Будет создана запись в прямой зоне test.ru, host2.test.ru соответствует 10.0.0.69.
- Будет создана MX запись в прямой зоне test.ru, host2.test.ru соответствует двум почтовым серверам mail1.test.ru (приоритет 10), mail2.test.ru (приоритет 20)

Создание PTR записи

Пример создания PTR записи. Сначала должна быть создана обратная зона 0.0.10.in-addr.arpa.

```
cl-dns-recadd -t ptr --ip 10.0.0.67 --host host.test.ru
```

- Будет создана запись в обратной зоне 0.0.10.in-addr.arpa, 10.0.0.67 соответствует host.test.ru

Создание CNAME записи

Пример создания CNAME записи. Сначала должна быть создана прямая зона test.ru.

```
cl-dns-recadd -t cname --host host.test.ru --cname calculate.ru
```

- Будет создана запись в прямой зоне test.ru, host.test.ru соответствует calculate.ru

Модификация записи

Для модификации DNS записи используется команда *cl-dns-recmod*.

Модификация A записи

Изменение доменного имени A записи и PTR записи

Пример.

```
cl-dns-recmod --host newname.test.ru oldname.test.ru
```

или

```
cl-dns-recmod --host newname.test.ru 10.0.0.5
```

Изменяет доменное имя oldname.test.ru на newname.test.ru

Исходные записи:

A запись, oldname.test.ru соответствует 10.0.0.5

PTR запись, 10.0.0.5 соответствует oldname.test.ru

Записи после модификации:

A запись, newname.test.ru соответствует 10.0.0.5

PTR запись, 10.0.0.5 соответствует newname.test.ru

Изменение ip A записи и PTR записи

Пример:

```
cl-dns-recmod --ip 10.0.0.6 10.0.0.5
```

или

```
cl-dns-recmod --ip 10.0.0.6 oldname.test.ru
```

Изменяет ip для доменного имени oldname.test.ru

Исходные записи:

A запись oldname.test.ru соответствует 10.0.0.5

PTR запись 10.0.0.5 соответствует oldname.test.ru

Записи после модификации:

A запись oldname.test.ru соответствует 10.0.0.6

PTR запись 10.0.0.6 соответствует oldname.test.ru

Изменение доменного имени A записи

Пример:

```
cl-dns-recmod --automod off --host newname.test.ru oldname.test.ru
```

или

```
cl-dns-recmod --automod off --host newname.test.ru 10.0.0.5
```

Изменяет доменное имя oldname.test.ru на newname.test.ru

Исходная запись:

А запись, oldname.test.ru соответствует 10.0.0.5

Запись после модификации:

А запись, newname.test.ru соответствует 10.0.0.5

Изменение ip A записи

Пример:

```
cl-dns-recmod --automod off --ip 10.0.0.6 10.0.0.5
```

или

```
cl-dns-recmod --ip 10.0.0.6 oldname.test.ru
```

Изменяет ip на 10.0.0.6 для доменного имени oldname.test.ru

Исходная запись:

А запись oldname.test.ru соответствует 10.0.0.5

Запись после модификации:

А запись oldname.test.ru соответствует 10.0.0.6

Модификация PTR записи

Изменение доменного имени PTR записи и A записи

Пример:

```
cl-dns-recmod -t ptr --host newname.test.ru oldname.test.ru
```

или

```
cl-dns-recmod -t ptr --host newname.test.ru 10.0.0.5
```

Изменяет доменное имя oldname.test.ru на newname.test.ru

Исходные записи:

PTR запись 10.0.0.5 соответствует oldname.test.ru

А запись oldname.test.ru соответствует 10.0.0.5

Записи после модификации:

PTR запись 10.0.0.5 соответствует newname.test.ru

А запись newname.test.ru соответствует 10.0.0.5

Изменение ip PTR записи и A записи

Пример:

```
cl-dns-recmod -t ptr --ip 10.0.0.6 10.0.0.5
```

или

```
cl-dns-recmod --ip 10.0.0.6 oldname.test.ru
```

Изменяет ip для доменного имени oldname.test.ru

Исходные записи:

PTR запись 10.0.0.5 соответствует oldname.test.ru

А запись oldname.test.ru соответствует 10.0.0.5

Записи после модификации:

PTR запись 10.0.0.6 соответствует oldname.test.ru

А запись oldname.test.ru соответствует 10.0.0.6

Изменение доменного имени PTR записи

Пример:

```
cl-dns-recmod -t ptr --automod off --host newname.test.ru oldname.test.ru
```

или

```
cl-dns-recmod -t ptr --automod off --host newname.test.ru 10.0.0.5
```

Изменяет доменное имя oldname.test.ru на newname.test.ru

Исходная запись:

PTR запись 10.0.0.5 соответствует oldname.test.ru

Запись после модификации:

PTR запись 10.0.0.5 соответствует newname.test.ru

Изменение ip PTR записи

Пример:

```
cl-dns-recmod -t ptr --automod off --ip 10.0.0.6 10.0.0.5
```

или

```
cl-dns-recmod -t ptr --ip 10.0.0.6 oldname.test.ru
```

Изменяет ip на 10.0.0.6 для доменного имени oldname.test.ru

Исходная запись:

PTR запись 10.0.0.5 соответствует oldname.test.ru

Запись после модификации:

PTR запись 10.0.0.6 соответствует oldname.test.ru

Модификация CNAME записи

Пример 1:

```
cl-dns-recmod --cname calculate.ru cn.test.ru
```

Изменяет CNAME запись.

Исходная запись:

CNAME запись cn.test.ru соответствует acoola.ru

Запись после модификации:

CNAME запись cn.test.ru соответствует calculate.ru

Пример 2:

```
cl-dns-recmod -t cname --host cname.test.ru cn.test.ru
```

Изменяет CNAME запись.

Исходная запись:

CNAME запись cn.test.ru соответствует calculate.ru

Запись после модификации:

CNAME запись cname.test.ru соответствует calculate.ru

Модификация или создание MX записи

Пример 1:

```
cl-dns-recmod --mx mail1.test.ru,mail2.test.ru test.test.ru
```

Заменяет а в случае отсутствия добавляет MX записи в A запись test.test.ru.

Исходная запись:

А запись test.test.ru - MX запись mail.test.ru (приоритет 10)

Запись после модификации:

А запись test.test.ru - MX запись mail1.test.ru (приоритет 10), MX запись mail2.test.ru (приоритет 20)

Пример 2:

```
cl-dns-recmod --mxmod mail2.test.ru,mailnew.test.ru test.test.ru
```

Изменяет MX запись.

Исходная запись:

А запись test.test.ru - MX запись mail1.test.ru (приоритет 10), MX запись mail2.test.ru (приоритет 20)

Запись после модификации:

А запись test.test.ru - MX запись mail1.test.ru (приоритет 10), MX запись mailnew.test.ru (приоритет 20)

Удаление DNS записи

Для удаления DNS записи используется команда *cl-dns-recdel*.

Удаление А или CNAME записи

Пример:

```
cl-dns-recdel --host test.test.ru
```

Будет удалена А или CNAME запись test.test.ru

Удаление PTR записи

Пример:

```
cl-dns-recdel --ip 10.0.0.20
```

Будет удалена PTR запись 20.0.0.10.in-addr.arpa (10.0.0.20 соответствует test.test.ru)

Удаление MX записи из А записи

Пример:

```
cl-dns-recdel --mx --host test.test.ru
```

Будет удалены все MX записи из А записи test.test.ru

Информация о DNS сервисе

Для получения информации о записях и зонах DNS сервиса используется команда *cl-info*.

Информация о всех зонах

```
cl-info -z dns
```

Информация о зоне

```
cl-info -Z <имя_зоны или сеть> dns
```

Примеры:

```
cl-info -Z 10.0.0.0/24 dns
```

Информация о обратной зоне 0.0.10.in-addr.arpa (сеть 10.0.0.0/24)

```
cl-info -Z test.ru dns
```

Информация о прямой зоне test.ru

Информация о записи

```
cl-info -r <имя_записи или ip> dns
```

Примеры:

```
cl-info -r 10.0.0.5 dns
```

Информация о записи в обратной зоне 5.0.0.10.in-addr.arpa (ip 10.0.0.5)

```
cl-info -r test.test.ru dns
```

Информация о записи в прямой зоне test.test.ru

Пример создания зоны и записей в ней

Необходимо создать зону test.ru а так-же доменные имена:

- test.ru - ip 10.0.0.1 - WEB сервер, DNS сервер.
- www.test.ru - ip 10.0.0.1 - WEB сервер (CNAME запись, тот же сервер что и test.ru)
- ftp.test.ru - ip 10.0.0.5, FTP сервер
- user1.test.ru - 10.0.0.100, компьютер пользователя

Для этого выполняем следующие команды после установки сервиса DNS:

1. Создаем зону test.ru с A записью (test.ru --> 10.0.0.1) и обратной зоной для сети 10.0.0.0/24

```
cl-dns-zoneadd -n test.ru --server test.ru --ipserver 10.0.0.1
```

1. Создаем CNAME запись (www.test.ru --> test.ru)

```
cl-dns-recadd -t cname --host www.test.ru --cname test.ru
```

1. Создаем A и PTR запись для FTP сервера

```
cl-dns-recadd --host ftp.test.ru --ip 10.0.0.5
```

1. Создаем A и PTR запись для компьютера пользователя

```
cl-dns-recadd --host user1.test.ru --ip 10.0.0.100
```

Проверка работоспособности DNS сервера

Для проверки работоспособности DNS сервера используйте утилиты nslookup или host.

После того как вы создали DNS зону и добавили в нее записи, нужно посмотреть существующие записи в зоне с помощью команды:

```
cl-info -Z имя_зоны dns
```

Пример:

Ранее была создана зона test.ru и записи в ней.

Выполняем:

```
cl-info -Z domain.ru dns
```

Результат:

```
Information about master DNS zone domain.ru
```

Field	Value
Zone name	domain.ru
Master authoritative server	domain.ru
NS record	domain.ru.
A record	10.0.0.5
Email administrator	root@domain.ru
Serial number	3
Refresh	8H
Update	2H
Expiry	2W
Minimum	2H

(10 rows)

```
Information about A records in master DNS zone domain.ru
```

Domain	ip
localhost.domain.ru	127.0.0.1
calculate.domain.ru	10.0.0.54

(2 rows)

Используя любую из существующих A записей проверьте работоспособность DNS сервера с помощью команд:

```
nslookup имя_A_записи ip_DNS_сервера
```

или

```
host имя_A_записи ip_DNS_сервера
```

Пример:

ip адрес проверяемого DNS сервера 10.0.0.5 информация о зоне domain.ru приведена в предыдущем примере.

Выполняем проверку при помощи *nslookup*:

```
nslookup calculate.domain.ru 10.0.0.5
```

Результат при нормальной работе сервиса DNS:

```
Server:      10.0.0.5
Address:     10.0.0.5#53
```

```
Name:   calculate.domain.ru
Address: 10.0.0.54
```

Выполняем проверку при помощи *host*:

```
host calculate.domain.ru 10.0.0.5
```

Результат при нормальной работе сервиса DNS:

Using domain server:

Name: 10.0.0.5

Address: 10.0.0.5#53

Aliases:

```
calculate.domain.ru has address 10.0.0.54
```

Настройка DHCP сервера

Установка DHCP сервиса в систему

Поддержка DHCP сервиса появилась в пакете [calculate-server](#) 2.1.4.

В качестве сервера используется dhcpcd.

Перед установкой

Убедитесь что у вас поднят сетевой интерфейс, интерфейсу присвоен ip находящийся в сети которая будет указана при установке сервиса DHCP (параметр командной строки установки сервиса --net).

Убедитесь что у вас в системе установлен сервис dns, если он не был установлен, выполните установку командой:

```
cl-setup dns
```

Установка DHCP сервиса выполняется командой

```
cl-setup --router <ip шлюза> --dnames <имена доменов> --range <диапазон ip> --net <ip сети с маской /24> --dnsip <ip DNS сервера> dhcp
```

Параметры:

- --router - ip адрес передаваемый клиентскому компьютеру в качестве шлюза по умолчанию
- --dnames - доменные имена передаваемые клиентскому компьютеру в качестве доменных имен для поиска, разделитель имен запятая. Первое имя используется для создания DNS домена на серверном компьютере в который будут добавлены компьютеры подключающиеся к сети.
- --range - диапазон адресов из которого клиентский компьютер получает свой ip, два ip адреса разделенные запятой.
- --net - сеть в которой будут находиться клиентские компьютеры
- --dnsip - ip адрес DNS сервера передаваемый клиентскому компьютеру

Примечание: для того чтобы клиентские компьютеры находили себя в DNS, параметр --dnsip должен указывать на текущий сервер с устанавливаемым сервисом DHCP

Пример (текущий сервер где устанавливаем DHCP сервис имеет ip 10.0.0.5):

```
cl-setup --router 10.0.0.1 --dnames domain.ru,domain.org --range 10.0.0.20,10.0.0.100 --net 10.0.0.0/24 --dnsip 10.0.0.5 dhcp
```

ВАЖНО: При установке сервиса DHCP изменяется доменное имя сервера на имя_хоста.первое_доменное_имя_для_поиска, если уже были настроены сервисы "FTP","SAMBA","MAIL", "PROXY", "JABBER" то необходимо первым доменным именем для поиска указать текущий домен, параметр "--dnames текущий_домен" (для его получения выполните команду hostname -d перед установкой сервиса). Второй вариант: необходимо в DNS прописать предыдущее полное имя сервера указывающее на его ip адрес)

Управление DHCP сервисом

Управление статическими хостами DHCP сервиса

Создание статического хоста DHCP

Для создания статического используется команда

```
cl-dhcp-hostadd --host <название хоста> --ip <ip адрес> --mac <mac адрес>
```

Параметры:

- --host - название клиентского компьютера, (название без домена)
- --ip - ip адрес который будет назначен клиентскому компьютеру
- --mac - mac адрес клиентского компьютера (можно узнать командой ifconfig)

Пример:

```
cl-dhcp-hostadd --host test --ip 10.0.0.20 --mac 00:17:31:c2:88:82
```

Модификация параметров статического хоста

Для модификации статического хоста используется команда

```
cl-dhcp-hostmod [параметры] название_хоста
```

Параметры:

- --ip - измененный ip адрес который будет назначен клиентскому компьютеру
- --mac - измененный mac адрес клиентского компьютера

Пример:

```
cl-dhcp-hostmod --ip 10.0.0.25 test
```

Удаление статического хоста

Для удаления статического хоста используется команда

```
cl-dhcp-hostdel --host <название хоста>
```

Параметры:

- --host - название клиентского компьютера, (название без домена)

Пример:

```
cl-dhcp-hostdel --host test
```

Управление сетями DHCP сервиса

Создание сети DHCP

На каждый сетевой интерфейс может быть создана только одна сеть.

Для создания сети используется команда

```
cl-dhcp-netadd --router <ip роутера> --dnames <имена доменов> --range <диапазон ip для динамических хостов> --net <ip сети с маской /24> --dnsip <ip DNS сервера> dhcp
```

Параметры:

- --router - ip адрес передаваемый клиентскому компьютеру в качестве шлюза по умолчанию

- `--dnames` - доменные имена передаваемые клиентскому компьютеру в качестве доменных имен для поиска, разделитель имен запятая. Первое имя используется для создания DNS домена на серверном компьютере в который будут добавлены компьютеры подключающиеся к сети.
- `--range` - диапазон адресов из которого клиентский компьютер получает свой ip, два ip адреса разделенные запятой.
- `--net` - сеть в которой будут находиться клиентские компьютеры
- `--dnsip` - ip адрес DNS сервера передаваемый клиентскому компьютеру

Пример:

```
cl-dhcp-netadd --router 10.0.0.1 --dnames domain.ru,domain.org --range 10.0.0.20,10.0.0.100 --net 10.0.0.0/24 --dnsip 10.0.0.5
```

Модификация параметров сети DHCP

Для модификации параметров сети используется команда

```
cl-dhcp-netmod [параметры] ip_и_маска_сети
```

Параметры:

- `--router` - измененный ip адрес передаваемый клиентскому компьютеру в качестве шлюза по умолчанию.
- `--dnames` - измененные доменные имена передаваемые клиентскому компьютеру в качестве доменных имен для поиска, разделитель имен запятая. Первое имя используется для создания DNS домена на серверном компьютере в который будут добавлены компьютеры подключающиеся к сети.
- `--range` - измененный диапазон динамических адресов которые раздаются клиентским компьютерам.
- `--dnsip` - измененный ip адрес DNS сервера передаваемый клиентскому компьютеру

Пример:

```
cl-dhcp-netmod --range 10.0.0.50,10.0.0.255 10.0.0.0/24  
cl-dhcp-netmod --router 10.0.0.1 10.0.0.0/24
```

Удаление сети DHCP

Если существует только одна сеть DHCP то ее удаление приведет к невозможности запуска DHCP сервера добавление новой сети вернет работоспособность.

Для удаления сети используется команда

```
cl-dhcp-netdel --net ip_и_маска_сети
```

Параметры:

- `--net` удаляемая сеть

Пример:

```
cl-dhcp-netdel --net 10.0.0.0/24
```

Информация о DHCP сервисе

Для получения информации о записях и зонах DNS сервиса используется команда `cl-info`.

Информация о всех сетях

```
cl-info -n dhcp
```

Информация о сети

```
cl-info -N <ip_и_маска_сети> dhcp
```

Пример.

```
cl-info -N 10.0.0.0/24 dhcp
```

Информация о сети 10.0.0.0/24

Информация о всех статических хостах==

```
cl-info --hosts dhcp
```

Информация о статическом хосте

```
cl-info -H <название хоста> dhcp
```

Пример.

```
cl-info -H computer dhcp
```

Информация о статическом хосте computer.

Настройка PXE

Введение

PXE --- среда для загрузки компьютеров с помощью сетевой карты без использования жёстких дисков, компакт-дисков и других устройств, применяемых при загрузке операционной системы. PXE-код, прописанный в сетевой карте, получает загрузчик из сети, после чего передаёт ему управление.

Для организации загрузки системы в PXE используются протоколы IP, UDP, DHCP и TFTP. Система, загружаемая по сети, является аналогом загрузки с liveCD.

Основные требования

Настройка PXE может быть выполнена на [Calculate Directory Server](#) с пакетом sys-apps/calculate-install версии 3.1.8 или выше.

Настройка

Перед настройкой PXE на сервере должен быть настроен [DHCP сервис](#).

Настройка PXE выполняется при помощи команды cl-install-pxe:

```
# установка iso образа  
cl-install-pxe --iso /var/calculate/linux/cld-13.6.2-i686.iso
```

Образ дистрибутива развертывается в каталог /var/calculate/pxe, настраиваются необходимые службы TFTP, NFS и DHCP. Поддерживается одновременное использование только одной версии дистрибутива.

После завершения установки все машины в сети смогут воспользоваться PXE загрузкой.

Настройка параметров по умолчанию

Вы можете изменить настройки загрузки PXE образа, такие как язык, часовой пояс, видео карта и другие при помощи параметра DEFAULTPARAM в файле /var/calculate/pxe/pxelinux.cfg/default. Запись должна быть добавлена после строки DEFAULT calcmenu.c32.

Поддерживаемые параметры

- Язык (для получения списка, выполните `cl-install -v --filter os_lang`) - `DEFAULTPARAM calculate=lang:ru_RU`
- Раскладка клавиатуры - `DEFAULTPARAM calculate=keymap:ru_RU`
- Часовой пояс - `DEFAULTPARAM calculate=timezone:Europe/Istanbul`
- Разрешение экрана - `DEFAULTPARAM calculate=resolution:2560x1600`
- Видео драйвер - `DEFAULTPARAM calculate=video:nvidia`
- Графическое ускорение (on/off) - `DEFAULTPARAM calculate=composite:on`
- Домен - `DEFAULTPARAM calculate=domain:192.168.2.3`
- Пароль для домена - `DEFAULTPARAM calculate=domain_pw:secret`

В качестве разделителя используется запятая. Пример: `DEFAULTPARAM calculate=domain:192.168.2.3, domain_pw:secret`

Примечание. Если указать только домен, пароль будет запрошен во время загрузки.

Настройка репликации Samba серверов

- [Настройка репликации Samba серверов](#)
- [Введение](#)
- [Настройка и запуск репликации](#)
- [Резервное копирование](#)
- [Настройка реплицирования сервисов](#)
- [Перенос настроек репликации на другие серверы](#)
- [Применение настроек репликации на других серверах](#)
- [Отключение репликации](#)
- [Поддержка со стороны клиента](#)

Введение

Часто возникает необходимость организации функционирования информационной системы с единой базой данных на нескольких серверах. Причиной этого становится, как правило, наличие на предприятии удаленных подразделений (территориально-распределенная структура предприятия). В этом случае приходится решать проблему организации передачи данных. Для решения данной проблемы используется *механизм репликации*.

Репликация --- механизм синхронизации содержимого нескольких копий объекта (база данных LDAP). Репликация --- это процесс, под которым понимается копирование данных из одного источника на множество других и наоборот.

При репликации изменения, сделанные в одной копии объекта, могут быть распространены в другие копии.

Репликация Samba подразумевает, то что на серверах, включенных в репликацию, будут одинаковые списки пользователей, и изменения, проведенные на одном из этих серверов, касающиеся учетных записей unix и samba, будут автоматически перенесены на остальные сервера.

Поддержка репликации LDAP включена в пакет calculate-server начиная с версии 2.0.7.

Настройка и запуск репликации

Резервное копирование

Перед тем как начинать настраивать репликацию, рекомендуется сделать резервную копию настроек, чтобы в случае некорректных действий не потерять данные:

`cl-backup`

Для восстановления данных из последней резервной копии используйте команду:

```
cl-backup -r
```

Настройка реплицирования сервисов

Добавление samba сервиса в репликацию и указание серверов производится командой
`cl-replication -r <полные имена серверов через запятую> samba`

Для получения полного имени сервера используйте команду `hostname -f`

При указании списка серверов репликации samba сервиса, для них автоматически добавляется репликация unix сервиса.

Перенос настроек репликации на другие серверы

Для того, чтобы перенести настройки репликации на другие серверы необходимо:

*на сервере, где настроена репликация, сделать резервную копию настроек:

```
cl-backup
```

*скопировать полученную резервную копию на все серверы, включенные в репликацию:

```
scp /var/calculate/server-backup/ldap/<последний бэкап>.tar.bz2  
root@otherserver:/var/calculate/server-backup/ldap/
```

Применение настроек репликации на других серверах

Для применения настроек репликации на других серверах необходимо выполнить на них команду

```
cl-rebuild --rep1
```

Отключение репликации

Для отключения репликации сервиса на сервере используется команда:

```
cl-replication --off <сервис>
```

Поддержка со стороны клиента

Репликацию поддерживает пакет calculate-client начиная с версии 2.0.13. Последовательность действий клиента:

1. при входе в сеанс клиент синхронизирует профиль с текущим сервером;
2. определяет включена ли репликация на сервере;
3. если включена репликация, считывает из LDAP информацию о местоположении последней копии профиля и при необходимости синхронизирует профиль пользователя с удаленным сервером;
4. при выходе из сеанса профиль записывается на текущий сервер;
5. в случае успешной синхронизации с удаленным сервером, делается пометка в LDAP записи (реплицируемой на все сервера), о новом расположении последней копии профиля.

Настройка репликации почтовых серверов

6. [Настройка репликации почтовых серверов](#)
7. [Определения](#)
8. [Схемы использования репликации](#)
9. [1. Почтовый домен без использования глобальной сети](#)
10. [2. Почтовый домен с использованием глобальной сети](#)
11. [3. Почтовый домен с почтовым релеем](#)
12. [Настройка и запуск репликации](#)
13. [Резервное копирование](#)

- [14. Настройка реплицирования сервисов](#)
- [15. Перенос настроек репликации на другие сервера](#)
- [16. Применение настроек репликации на других серверах](#)
- [17. Пример создания почтового ящика](#)
- [18. Описание реализации](#)

Поддержка почтовой репликации включена в пакет [calculate-server](#) начиная с версии 2.0.9.

Определения

Инtranet - в отличие от сети интернет, это внутренняя частная сеть организации. Как правило, инtranet --- это интернет в миниатюре, который построен на использовании протокола IP для обмена и совместного использования некоторой части информации внутри этой организации.

Почтовый домен - это совокупность почтовых ящиков, групп и списков рассылки, идентифицируемая единым доменным именем. Это имя называется основным именем почтового домена. Все почтовые ящики, группы и списки рассылки, относящиеся к одному почтовому домену имеют в адресе электронной почты общую часть после символа "@".

Репликация почтовых серверов - служит для возможности создания почтовых доменов внутри сети инtranet, состоящей из нескольких почтовых серверов. Серверы, участвующие в репликации, могут выступать в роли почтового сервера либо почтового релея.

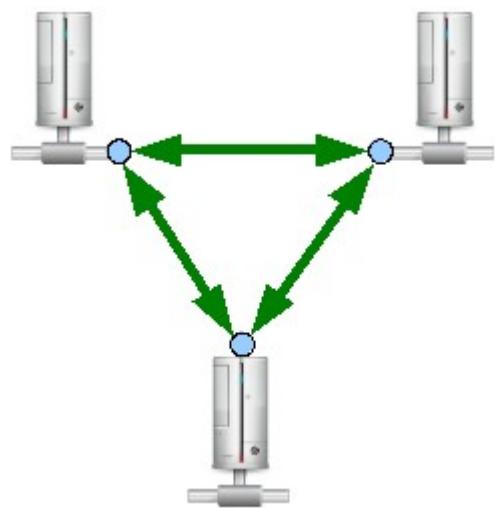
Почтовый релей --- сервер, занимающийся получением/пересылкой электронной почты. Релей обеспечивает прием письма, временное хранение (от нескольких минут до недели), пересылку сообщения почтовому серверу (или следующему почтовому релею).

Почтовый сервер - сервер, который может выполнять не только функции почтового релея, но также хранит пользовательскую почту и предоставляет доступ к этой почте по протоколам POP3 и/или IMAP.

Схемы использования репликации

Ниже приведены три схемы настройки репликации почтовых серверов. На приведенных схемах, синими стрелками обозначена входящая из интернета почта, фиолетовыми - исходящая в интернет, зелеными - пересылка почты внутри сети инtranet.

1. Почтовый домен без использования глобальной сети



Описание

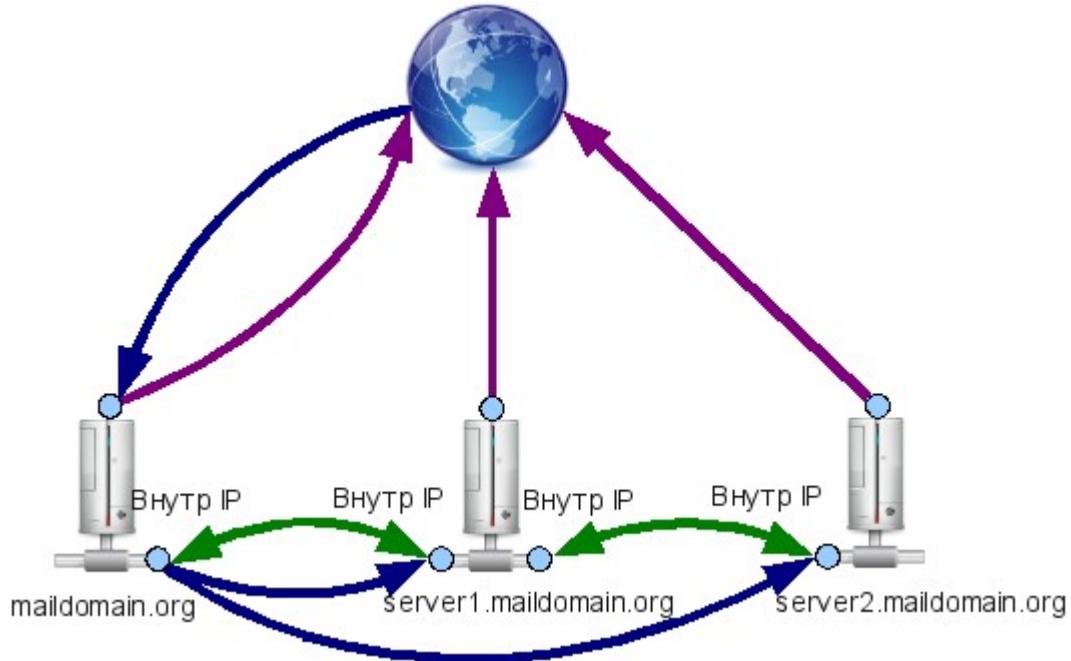
В данной схеме почтовые серверы осуществляют пересылку почты только в пределах сети инtranet. Почтовые сервера объединены в виртуальный почтовый домен (на примере: *maildomain.org*), благодаря чему все почтовые адреса выглядят как: "<имя почтового ящика>@*maildomain.org*".

Настройка схемы

Для настройки заданной схемы:

- выполните настройку на одном из почтовых серверов ([Unix](#) и [Samba](#)) сервисов.
- настройте репликацию Unix и Mail сервисов, указав полные имена всех почтовых серверов, участвующих в репликации.
- установите настройки репликации на остальные почтовые серверы участвующие в репликации.

2. Почтовый домен с использованием глобальной сети



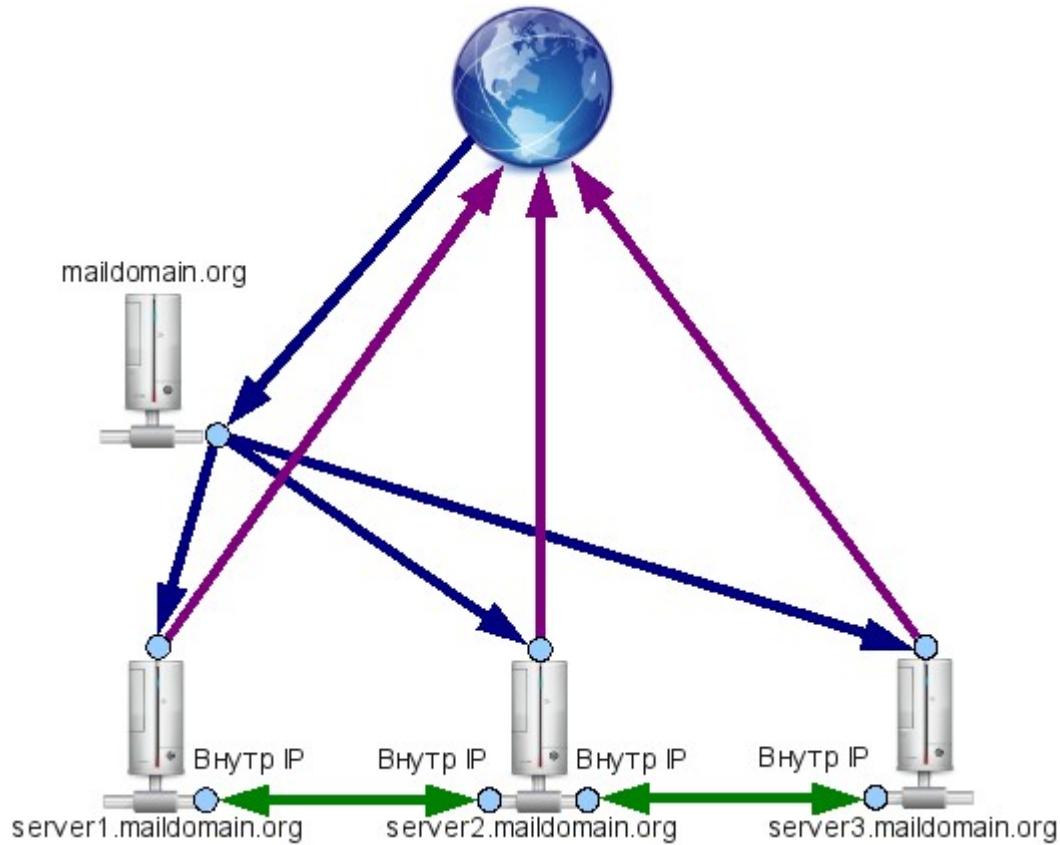
Описание

В отличие от первой схемы, здесь один из серверов принимает все письма, приходящие с внешних адресов, осуществляя доставку другим почтовым серверам внутри интранет. Каждый почтовый сервер должен иметь реальный IP адрес для отправки писем внешним адресатам. Для такого сервера должна присутствовать [MX запись](#) в DNS.

Настройка схемы

Настройка заданной схемы выполняется аналогично первой схеме.

3. Почтовый домен с почтовым релеем



Описание

Особенностью схемы является наличие почтового реля, который берет на себя функции переадресации почты приходящей с внешних адресов. Благодаря репликации, почтовый сервер всегда знает хосты серверов всех почтовых ящиков обслуживаемых доменов.

Отправка писем во внешние адреса осуществляется каждым сервером минуя почтовый релей.

Настройка схемы

Для настройки заданной схемы:

- выполните настройку на одном из почтовых серверов ([Unix](#) и [Samba](#)) сервисов.
- настройте репликацию Unix и Mail сервисов, указав полные имена всех почтовых серверов участвующих в репликации, за исключением почтового реля, на котором репликация Unix сервиса будет отсутствовать.
- установите настройки репликации на остальные почтовые серверы участвующие в репликации.

Настройка и запуск репликации

Резервное копирование

Перед тем как начинать настраивать почтовую репликацию, рекомендуется сделать резервную копию настроек, чтобы в случае некорректных действий не потерять данные:

```
c1-backup
```

Для восстановления данных из последней резервной копии используйте команду:

```
c1-backup -r
```

Настройка реплицирования сервисов

Добавление mail сервиса в репликацию и указание серверов производится командой

```
cl-replication -r <полные имена серверов через запятую> mail
```

Для получения полного имени сервера используйте команду:

```
hostname -f
```

При репликации только mail сервиса все перечисленные серверы будут почтовыми релеями. Для того чтобы сделать их (все или некоторые) почтовыми серверами, необходимо добавить их в репликацию unix сервиса. Это делается командой:

```
cl-replication -r <полные имена серверов через запятую> unix
```

Перенос настроек репликации на другие сервера

Для того, чтобы перенести настройки репликации на другие серверы необходимо:

- на сервере, где настроена репликация, сделать резервную копию настроек:

```
cl-backup
```

- скопировать полученную резервную копию на все серверы, включенные в репликацию:

```
scp /var/calculate/server-backup/ldap/<последний бэкап>.tar.bz2  
root@otherserver:/var/calculate/server-backup/ldap/
```

Применение настроек репликации на других серверах

Для применения настроек репликации на других серверах необходимо выполнить на них команду

```
cl-rebuild --repl
```

Пример создания почтового ящика

После настройки репликации серверов, вы можете создавать пользователей на любом из почтовых серверов. Для этого используйте команду cl-useradd с параметром "-e" (альтернативный почтовый адрес), где в качестве параметра следует указывать почтовый домен.

Пример:

```
cl-useradd -e user1@maildomain.org user1 mail
```

Описание реализации

Репликация mail сервиса заключается в том, что реплицируются только почтовые алиасы.

Алиас --- запись (в данном случае в LDAP), которая говорит о том: какие альтернативные почтовые адреса соответствуют реальным почтовым адресам (в данном случае один реальный почтовый адрес).

При отправке письма почтовый сервер просматривает по LDAP: принадлежит ли адресат письма текущему почтовому домену. Если принадлежит, то по записи LDAP определяется почтовый сервер получатель этого письма, и письмо отправляется на определенный почтовый сервер в интранете. Если же адресат не принадлежит текущему почтовому домену, то письмо отправляется в интернет.

Настройка сервера шлюза

- [Настройка сервера шлюза](#)
- [Используемый программный пакет](#)
- [Программные пакеты расширяющие функционал](#)
- [Настройки файрвола](#)
- [Объявление зон](#)
- [Связывание интерфейса с определенной зоной](#)

- [Связывание IP-адресов с определенной зоной](#)
- [Задание действий для пакетов по умолчанию](#)
- [Добавление маскарадинга](#)
- [Добавление правил](#)
- [Определение туннелей](#)
- [Добавление запуска Shorewall при загрузке](#)
- [Настройка управления пропускной способностью](#)
- [Включение управления трафиком на интерфейсе.](#)
- [Определение классов для трафика](#)
- [Настройка маркировки трафика](#)
- [Примеры настроек](#)
- [Пример настройки шлюза](#)
- [Добавление ip телефонии](#)
- [IPSEC туннели](#)
- [Пример управления трафиком](#)

Для настройки Calculate Directory Server в качестве шлюза используется Shorewall. Shorewall или более точно Shoreline Firewall --- инструмент для настройки файрвола (межсетевого экрана). Технически является надстройкой над подсистемой Netfilter (iptables/ipchains) ядра Linux и обеспечивает упрощённые методы конфигурирования данной подсистемы. Он предоставляет более высокий уровень абстракции для описания правил работы файрвола.

Программа не является демоном, то есть не работает постоянно. Правила хранятся в текстовых файлах, при запуске shorewall считывает свои файлы конфигурации и преобразует их в настройки понятные ipchains/iptables, после чего данные настройки файрвола могут действовать до перезапуска операционной системы.

Используемый программный пакет

net-firewall/shorewall

Для включения примеров и документации необходимо собрать пакет с USE флагом "doc".

Программные пакеты расширяющие функционал

- **net-firewall/xtables-addons** - расширения не добавленные в kernel/iptables (для определения p2p трафика)
- **net-misc/17-filter-userspace** - классификация пакетов по содержимому (для определения трафика по содержимому)
- **net-misc/linux-igd** - для добавления функционала UPnP

Настройки файрвола

- **Зоны**

Shorewall видит сеть, в которой он работает, как состоящую из набора зон (zones). Поэтому настройка начинается с определения одной или нескольких зон в файле **/etc/shorewall/zones**. Зоны представляют собой отдельные ip адреса хостов, подсети, или входящие/исходящие пакеты для конкретного интерфейса. Они могут относиться к внешней сети, внутренней сети и [DMZ](#)).

- **Интерфейсы**

Зоны распознаются либо по подключенному к ним сетевому интерфейсу, определенному в файле **/etc/shorewall/interfaces**, либо по IP-адресу подсети, указанному в файле **/etc/shorewall/hosts**. У одной зоны может быть несколько интерфейсов, а у одного интерфейса -- несколько зон. Заметим, что Shorewall рассматривает систему файрвола как свою собственную зону.

- **Действия**

Определив зоны, нужно задать действие по умолчанию в файле `/etc/shorewall/policy` (например, ACCEPT или DROP), применяемую к трафику между каждой исходной зоной и зоной назначения.

- **Политики**

Наконец, в файле `/etc/shorewall/rules` определяются подробности исключений из политики, разрешающие доступ к заданным портам и т.д.

Объявление зон

Объявление зон производится в файле `/etc/shorewall/zones`. Для того, чтобы объявить зону необходимо ее название (ZONE) и ее тип (TYPE) (`firewall` зона файрвола, `ipv4` - стандартная зона).

Порядок описания зон важен, так как именно в таком порядке будут вызываться цепочки `iptables`, проверяющие пакеты направленный из одной зоны в другую. То есть если поступает пакет адресованный файрволу, то в соответствии с вышеописанным примером пакет вначале проверяется на принадлежность зоне `net`, а затем уже зоне `loc`. Это правило является важным если одна из зон включает в себя другую. Например в зоне `loc` можно создать зону `phone`, которая представляет собой несколько ip адресов из зоны `loc`. Таким образом если вначале описать зону `loc`, а затем `phone`, то получится, что политика описанная для зоны `phone` никогда не будет выполняться, так как трафик будет попадать в цепочки для `loc`. Данная проблема может быть решена указанием зоны `phone` как подзоны `loc` (`phone : loc ipv4`). Пример настройки с двумя сетевыми картами:

- сам маршрутизатор (`fw`) обозначается как `firewall`
- интернет (`net`) использует `ipv4`
- локальная сеть (`loc`) использует `ipv4`

#ZONE	TYPE	OPTIONS	IN OPTIONS	OUT OPTIONS
#				
fw	firewall			
net	ipv4			
loc	ipv4			

Более подробно о настройке этого файла можно прочитать, выполнив `man shorewall-zones`.

Связывание интерфейса с определенной зоной

Для связывания интерфейса с определенный зоной используется файл `/etc/shorewall/interfaces`, так же в этом файле можно указать какими фильтрами будут обрабатываться входящие пакеты.

Чтобы назначить интерфейс для определенной зоны, необходимо указать название этой зоны (ZONE), сам интерфейса (INTERFACE), широковещательный адрес (BROADCAST) (рекомендуется указать `detect`), а также при необходимости перечислить фильтры (OPTIONS).

Пример с двумя сетевыми картами:

В файле указываем, что `eth0` выходит в интернет (`net`), `eth1` в локальную сеть (`loc`). `tcpflags`, `routefilter`, `nosmurf`, `logmartians` - применяем фильтры для входящего трафика.

- `tcpflags` - пакеты полученные этим интерфейсом проверяются на некорректное сочетание TCP флагов;
- `routefilter` - добавить anti-spoofing проверку (один из видов атак);
- `nosmurf` - не пропускать пакет с широковещательным исходящим адресом;
- `logmartians` - логирование пакетов, которые содержат невозможный исходящий адрес.

#####
#####

```
#ZONE INTERFACE      BROADCAST      OPTIONS
net    eth0          detect
tcpflags,routefilter,nosmurfs,logmartians
loc    eth1          detect
tcpflags,routefilter,nosmurfs,logmartians
```

Начиная с версии 4.5.3 shorewall поддерживает второй формат в котором нет необходимости указывать BROADCAST

```
?FORMAT 2
#####
#####
#ZONE INTERFACE      OPTIONS
net    eth0          tcpflags,routefilter,nosmurfs,logmartians
loc    eth1          tcpflags,routefilter,nosmurfs,logmartians
```

Более подробно о настройке этого файла можно прочитать, выполнив man shorewall-interfaces.

Связывание IP-адресов с определенной зоной

Для связывания IP-адресов или подсетей с зоной используется файл /etc/shorewall/hosts. Для связывания хостов/подсетей с зоной необходимо указать название зоны (ZONE), и сетевой интерфейс с подсетью или ip адресами (HOST).

Пример определения vpn зоны (текущий шлюз устанавливает туннель с другим шлюзом, который подключен к сегменту 10.0.0.0/24).

```
#####
#####
#ZONE           HOST(S)      OPTIONS
vpn            eth0:10.0.0.0/24
```

Пример выделения ip адресов для телефонов (множество ip адресов 192.168.0.60-192.168.0.80) в отдельную зону phone. Это может быть удобно для управлением доступом политиками, а не исключениями.

```
#####
#####
#ZONE           HOST(S)      OPTIONS
phone          eth1:192.168.0.60-192.168.0.80
```

Более подробно о настройке этого файла можно прочитать, выполнив man shorewall-hosts.

Задание действий для пакетов по умолчанию

Действие по умолчанию (политика), применяемое к трафику между каждой исходной зоной и зоной назначения указывается в файле /etc/shorewall/policy. Правило описывается следующим образом: исходная зона (SOURCE), зона назначения (DEST), правило по умолчанию (POLICY) (DROP,ACCEPT,REJECT), и если необходимо уровень логирования (например info).

Политика рассматривает пакеты, которые создают новые соединения. **RELATED/ESTABLISHED** пакеты (пакеты установленного соединения) по умолчанию пропускаются. Для примера установим следующие политики:

- разрешить доступ из локальной сети в интернет
- разрешить доступ из фаервола во всходу
- сбрасывать все пакеты пришедшие из интернета + вести лог
- остальные пакеты запрещать с сообщением об ошибке + вести лог

```
#####
#SOURCE DEST POLICY LOG LIMIT: CONNLIMIT:
```

LEVEL BURST MASK

```
loc net ACCEPT $FW all ACCEPT net all DROP info all all REJECT info
```

Более подробно о настройке этого файла можно прочитать, выполнив `man shorewall-policy`.

Добавление маскарадинга

Маскарадинг --- тип трансляции сетевого адреса, при которой адрес отправителя подставляется динамически, в зависимости от назначенного интерфейсу адреса.

Определение dynamic NAT (Masquerading) и Source NAT (SNAT) производится в `/etc/shorewall/masq`.

Для добавления маскарадинга к конкретному интерфейса необходимо его указать (INTERFACE:DEST), затем указать подсеть, для которой будет применяться данный маскарадинг (SOURCE), исходящий адрес (ADDRESS) для SNAT. Если исходящий адрес не указывать, то будет использоваться dynamic NAT.

Для примера укажем, что для всех пакетов уходящие на eth0 с нашей внутренней сети (192.168.0.0/24) применять SNAT (24.56.78.21).

```
#####
####
#INTERFACE:DEST      SOURCE          ADDRESS        PROTO    PORT(S)
IPSEC   MARK    USER/
#
GROUP
eth0           192.168.0.0/24  24.56.78.21
```

При необходимости можно исключить NAT при отправке пакетов в некоторые подсети, например для настройки ipsec туннеля.

Для примера отключим NAT при отправки пакета в подсеть 192.168.2.0/24:

```
eth0:!192.168.2.0/24  192.168.0.0/24  24.56.78.21
```

Более подробно о настройке этого файла можно прочитать, выполнив `man shorewall-masq`.

Добавление правил

Правила необходимы для описания исключений из политик, применяемых к трафику. Правила помещаются в файл `/etc/shorewall/rules`.

Файл содержит три секции: RELATED,ESTABLISHED,NEW, соответствующие критерий состояния соединения. Критерий определяется при помощи критерия conntrack, который можете классифицировать пакеты на основании их отношения к соединениям. В частности, состояние NEW позволяет выделять только пакеты, открывающие новые соединения, состояние ESTABLISHED --- пакеты, принадлежащие к установленным соединениям, состоянию RELATED соответствуют пакеты, открывающие новые соединения, логически связанные с уже установленными (например, соединение данных в пассивном режиме FTP). Состояние INVALID означает, что принадлежность пакета к соединению установить не удалось (INVALID также относится к секции NEW). Как уже описывалось ранее политика определяет действие для секции NEW, к RELATED и ESTABLISHED по умолчанию применяется ACCEPT.

Ниже представленные примеры записываются в секцию NEW.

Запрещение выхода в интернет с определенных узлов

#ACTION	SOURCE	DEST	PROTO	DEST
SOURCE	ORIGINAL	RATE	USER/	MARK
TIME				CONNLIMIT
#				
PORT(S)	DEST	LIMIT	GROUP	PORT
REJECT	loc:10.0.0.100-10.0.0.254	net		

Перенаправление порта (DNAT) 80.

#ACTION	SOURCE	DEST	PROTO	DEST	SOURCE
ORIGINAL	RATE	USER/	MARK	CONNLIMIT	TIME
DEST	LIMIT	GROUP		PORT	PORT(S)
#					
HTTP/DNAT	net	loc:10.0.0.5			

Если нужно, чтобы для пакетов отправленных из локальной сети на внешний ip шлюза также применялось это правило, то необходимо отредактировать файлы masq и interfaces

В interfaces добавить опцию routeback для loc сети, разрешающее маршрутизатору пересыпать пакеты из локальной сети в локальную

```
loc eth1 detect tcpflags,routeback
```

В masq добавить правило SNAT, чтобы пакет для 10.0.0.5 был отправлен от ip шлюза (10.0.0.1 - внутренний ip адрес шлюза)

```
eth1:10.0.0.5 eth1 10.0.0.1
```

К выше добавленному правилу в rules добавить правило DNAT что бы все пакеты из локальной сети отправлялись на 10.0.0.5, (1.2.3.4 - внешний ip адрес шлюза помещаем в ORIGINAL DEST)

```
HTTP/DNAT loc loc:10.0.0.5 - - 1.2.3.4
```

Перенаправление порта 222 на 22

#ACTION	SOURCE	DEST	PROTO	DEST	SOURCE
ORIGINAL	RATE	USER/	MARK	CONNLIMIT	TIME
DEST	LIMIT	GROUP		PORT	PORT(S)
#					
DNAT	net	loc:10.0.0.5:22	tcp	222	

Перенаправление LDAP порта, например для возможности выполнения репликации сервисов unix,samba,mail на Calculate Directory Server через интернет

#ACTION	SOURCE	DEST	PROTO	DEST	SOURCE
ORIGINAL	RATE	USER/	MARK	CONNLIMIT	TIME
DEST	LIMIT	GROUP		PORT	PORT(S)
#					
LDAP/DNAT	net	loc:10.0.0.5			

В примере с 80 портом использован макрос, который представляет собой шаблон для создания одного или нескольких правил. Стандартные макросы находятся в /usr/share/shorewall и называются macro.имя_макроса. Содержимое макроса HTTP.

```
#ACTION SOURCE DEST PROTO DEST SOURCE RATE USER/
```

#				PORT(S)	PORT(S)	LIMIT	GROUP
PARAM	-	-	tcp	80			

В соответствии с содержимым макроса, его использование эквивалентно

#ACTION ORIGINAL	SOURCE RATE	DEST USER/	PROTO	DEST CONNLIMIT	SOURCE TIME	
# DEST	LIMIT	MARK		PORT	PORT(S)	
DNAT	net	GROUP loc:10.0.0.5	tcp	80		

Отправитель и получатель пакета может быть задан следующим образом dmz:192.168.2.2 узел 192.168.2.2 в DMZ зоне

net:155.186.235.0/24 подсеть 155.186.235.0/24 в интернете

loc:192.168.1.1,192.168.1.2 узлы 192.168.1.1 и 192.168.1.2 в локальной сети

loc:~00-A0-C9-15-39-78 узел в локальной сети с MAC адресом 00:A0:C9:15:39:78

net:192.0.2.11-192.0.2.17 узлы в диапазоне 192.0.2.11-192.0.2.17 в интернете

net:!192.0.2.11-192.0.2.17 все узлы из интернета исключая 192.0.2.11-192.0.2.17

net:155.186.235.0/24!155.186.235.16/28 подсеть 155.186.235.0/24 в интернете исключая 155.186.235.16/28

Более подробно о настройке этого файла можно прочитать, выполнив man shorewall-rules.

Определение туннелей

Для того чтобы Shorewall не сбрасывал шифрованный трафик - необходимо описать туннель в /etc/shorewall/tunnels. Этот файл используется для определения правил пропуска инкапсулированного (обычно шифрованного) трафика между Shorewall и удаленным шлюзом.

В описании указывается тип туннеля (TYPE), зона из которой приходит трафик туннеля(ZONE), адрес удаленного шлюза (GATEWAY).

Для примера определим IPSEC туннель (ESP) между текущим шлюзом и удаленным (4.33.99.124).

#TYPE	ZONE	GATEWAY
ipsec:esp	net	4.33.99.124

Более подробно о настройке этого файла можно прочитать, выполнив man shorewall-tunnels.

Добавление запуска Shorewall при загрузке

Для того, чтобы Shorewall запускался при загрузке шлюза, необходимо выполнить:

rc-update add shorewall boot

Настройка управления пропускной способностью

Организация очереди (очередизация) определяет способ отсылки данных. Важно понимать, что мы можем управлять лишь скоростью передачи отправляемых данных.

В том виде, в каком сейчас существует Internet, мы не можем контролировать объем входящего трафика. Это что-то вроде почтового ящика (не электронного!). Нет никакого способа влиять на то, какой объем почты приходит к вам, разве что общаясь с каждым респондентом.

Однако, Internet, в большинстве своем, основан на протоколе TCP/IP, а у него есть несколько свойств, которые могут нам помочь. TCP/IP не может узнать пропускной способности сети между двумя хостами, поэтому он начинает передавать данные все быстрее и быстрее (это называется "медленный старт"). Когда пакеты начинают теряться из-за перегрузки передающей среды, передача тормозится. На самом деле все немного сложнее и умнее, но об этом позже.

Дисциплины обработки очереди определяют способ передачи данных. Существуют бесклассовые дисциплины и полноклассовые. Они в общем случае, получают данные, переупорядочивают, вносят задержку или уничтожают их. Полноклассовые позволяют при управлении использовать классы для конкретного вида трафика. Таким образом для интерфейса существует корневая дисциплина, которая может быть либо полноклассовой, либо бесклассовой. Полноклассовая в свою очередь содержит множество классов, позволяющих управлять трафиком.

Для того, чтобы управлять трафиком необходимо определить корневую дисциплину применяемую к устройству.

Входящий трафик подается во входящую дисциплину, которая может содержать ряд фильтров, посредством которых отдельные пакеты могут быть отброшены (потеряны, отфильтрованы). Это называется Ограничением (Policing).

Все это происходит на самой ранней стадии, прежде чем пакет будет передан для дальнейшей обработки. Таким образом достигается уменьшение нагрузки на центральный процессор.

Если пакет благополучно миновал эту стадию, то далее он может быть передан либо локальным приложениям (в этом случае он попадает в стек IP для дальнейшей обработки), либо в сеть, через исходящий классификатор, на другой узел сети. Пользовательские приложения так же могут отправлять данные в сеть, которые аналогичным образом движутся через исходящий классификатор. Исходящий классификатор "разбивает" трафик по очередям, каждая из которых имеет свою дисциплину организации. В случае конфигурации по-умолчанию имеется единственная исходящая очередь -- `pfifo_fast`. Этот процесс называется "постановкой в очередь".

Попав в соответствующую очередь, пакет ожидает, пока ядро передаст его сетевому интерфейсу. Этот процесс называется "выборка из очереди".

Включение управления трафиком на интерфейсе.

Для включения управления трафиком используется файл `/etc/shorewall/tcdevices`.

Записи в этом файле определяют ширину канала для интерфейса, на котором вы хотите использовать управление пропускной способностью. Ширина канала может измеряться в kbps, mbps, kbit, mbit и bps.

`IN-BANDWIDTH` содержит ширину входящего канала для интерфейса. Необходимо учитывать, что невозможно управлять входящим трафиком, так как трафик уже получен, но можно указать максимальное количество получаемого трафика, при этом пакеты превышающие это количество будут **брошены**. Если вы не хотите сбрасывать трафик - установите ноль или прочерк.

`OUT-BANDWIDTH` содержит ширину исходящего канала для интерфейса. Это максимальная скорость используется как "full" при классификации трафика. Исходящий трафик превышающий максимальный **сбрасывается**.

Если `shorewall` устанавливается на шлюз в котором важно управлять транзитным трафиком, то для обоих интерфейсов необходимо не указывать входящий трафик, так как возможны проблемы в дальнейшем при управлении пропускной способностью.

Пример настройки канала 10мбит на шлюзе (входящий трафик из интернета управляется как исходящий в локальную сеть):

#NUMBER:	IN-BANDWITH	OUT-BANDWIDTH	OPTIONS	REDIRECTED INTERFACES
#INTERFACE				
eth0	-	10mbit		
eth1	-	10mbit		

Более подробно о настройке этого файла можно прочитать, выполнив `man shorewall-tcdevices`.

Определение классов для трафика

Определения классов производится в `/etc/shorewall/tcclasses`. В каждой строке определяется класс трафика, а именно: интерфейс для которого создается класс (INTERFACE), маркер принадлежности трафика к классу (MARK), гарантированная пропускная способность (RATE), желаемая пропускная способность (CEIL), приоритет (PRIORITY), дополнительные параметры (OPTIONS).

Маркер будет использоваться при настройки правил (`tcrules`) принадлежности трафика определенному классу. *Гарантированная и желаемая пропускная способность* может быть указана в абсолютных величинах или относительно полной пропускной способности указанной для интерфейса (например `2*full/3`, две трети канала). *Гарантированная пропускная способность* означает, что при пиковых нагрузках данный вид трафика будет иметь как минимум эту пропускную способность, *желаемая* - что пропускная способность для данного вида трафика не будет больше указанной величины. Чем *меньше* значение *приоритета*, тем *выше* приоритет для трафика. Пока не будет обработан трафик с высшим приоритетом, трафик с меньшим приоритетом не обрабатывается. Дополнительные параметры перечисляются через запятую и могут принимать следующие значения:

- `default` - класс по умолчанию, не маркованный трафик будет отнесен в этот класс.
- `tos=0xvalue [/0xmask]` - классификация трафика на основе TOS байта.
- `tos=tosname` - так же классификация трафика по TOS байту на основании пяти стандартных значений
- `tcp-ack` - служит для определения tcp ask пакетов, имеющих размер менее 64 байт.
- `pfifo` - использовать для класса дисциплину pfifo, вместо SFQ.
- `limit=number` - определяет максимальное число пакетов которые могут быть помещены в эту очередь. *Ниже в примере* настраивается классификация следующим образом:
 - 100-180kbit для трафика с `tos` байтом 68/b8 (EF и AFF3-1 трафик VOIP устройств)
 - Интерактивный трафик и TCP acks и любой пакет с маркером 2 будет иметь гарантированную 1/4 от всей полосы и может занимать весь канал.
 - Неклассифицированный трафик и пакет с маркером 3 будет иметь гарантированную 1/4 от всей полосы и может занимать весь канал.
 - Пакет с маркером 4 имеют наименьший приоритет гарантированную 1/8 часть полосы, и может занимать 80% от всего канала.

```
#INTERFACE MARK RATE CEIL PRIORITY OPTIONS eth0 1 100kbit 180kbit 1
tos=0x68/0xfc,tos=0xb8/0xfc eth0 2 full/4 full 2 tcp-ack,tos-minimize-delay eth0 3 full/4 full 3 default
eth0 4 full/8 full*8/10 4
```

Более подробно о настройке этого файла можно прочитать, выполнив `man shorewall-tcclasses`.

Настройка маркировки трафика

Настройка маркировки осуществляется в файле `/etc/shorewall/mangle`. Записи в этом файле при помощи маркировки определяют принадлежность пакета тому или иному классу. Для определения условия установки маркера используются следующие поля:

- ACTION - действие
- SOURCE - адрес отправителя пакета
- DEST - адрес получателя пакета
- PROTO - протокол
- DEST PORT - порт или порты получателя
- SOURCE PORT - порт или порты отправителя
- USER - пользователь, которому принадлежит процесс, создавший пакет (имеет смысл только для исходящего трафика с самого файервола)
- TEST - проверка установленного маркера

- LENGTH - размер пакета
- TOS - значение TOS байта
- CONNBYTES - диапазон передаваемых данных для соединения
- HELPER - имя Netfilter helper модуля, например для определения ftp, sip, amanda, и т.д. Для указания маркера используется следующие значения:
- MARK(значение) [:{C|F|P|T|CF|CP|CT}]
 - F - маркировка производится в FORWARD цепочке
 - P - маркировка производится в PREROUTING цепочке
 - T - маркировка производится в POSTROUTING цепочке
 - CF - маркировка соединения производится в FORWARD цепочке
 - CP - маркировка соединения производится в PREROUTING цепочке
 - CT - маркировка соединения производится в POSTROUTING цепочке
- RESTORE[/mask] - восстановить маркировку пакета из маркировки соединения
- SAVE[/mask] - сохранить маркировку пакета в маркировке соединения
- CONTINUE - не обрабатывать последующие правила в таблице маркировки

Пример

- Все пакеты GRE (протокол 47), не созданные в системе файрвола и имеющие целевой адрес 155.186.235.151, должны иметь метку 1.
 - Все пакеты SSH, приходящие с 192.168.1.0/24 и предназначенные для 155.186.235.151, должны иметь метку 2.
 - Все пакеты P2P промаркировать меткой 4.
- ```
#ACTION SOURCE DESTINATION PROTOCOL PORT(S) SPORT USER TEST MARK(1):T
0.0.0.0/0 155.182.235.151 47 MARK(2):T 192.168.1.0/24 155.182.235.151 tcp 22 RESTORE 0.0.0.0/0
0.0.0.0/0 all - - - 0 CONTINUE 0.0.0.0/0 0.0.0.0/0 all - - - !0 MARK(4):T 0.0.0.0/0 0.0.0.0/0 ipp2p:all
SAVE 0.0.0.0/0 0.0.0.0/0 all - - - !0
```

Последние четыре правила означают следующее:

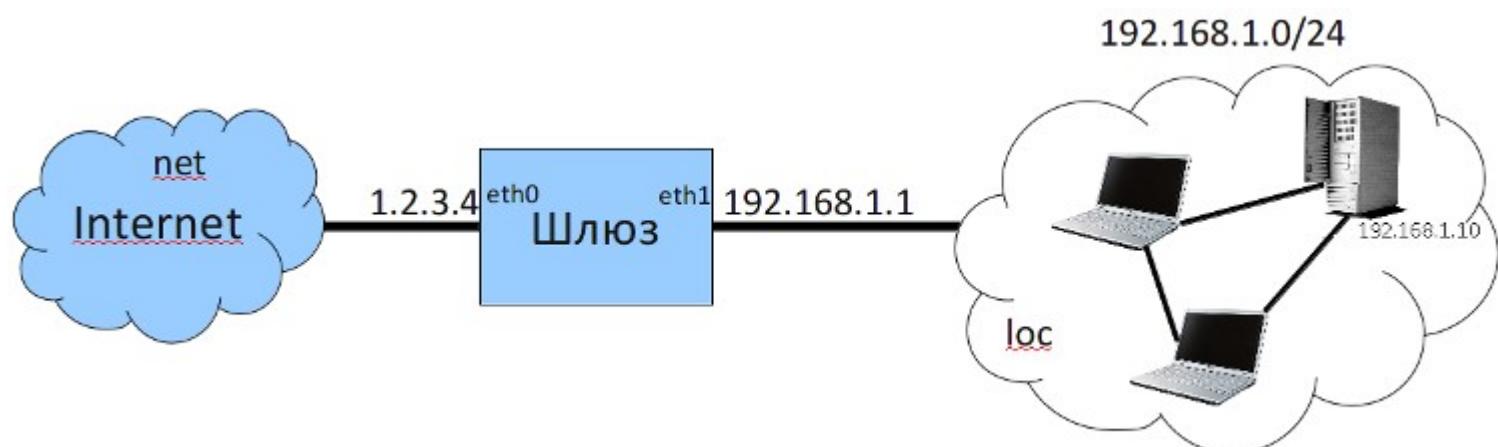
Если пакет не был классифицирован (метка пакета 0), то скопировать метку соединения в метку пакета. Если метка пакета уже задана, то никаких действий более не требуется. Если пакет относится к типу P2P, то задать метку пакета 4. Если метка пакета задана, то сохранить ее в качестве метки соединения.

Справка по настройке man shorewall-tcrules.

## Примеры настроек

### Пример настройки шлюза

Допустим у нас есть сеть представленная на рисунке ниже:



- компьютеры внутренней сети имеют адреса 192.168.1.0/24

- Calculate Directory Server, подключенный к внутренней сети через интерфейс eth1 с установленным ip адресом 192.168.1.1, и имеющий выход в интернет через интерфейс eth0 с установленным ip адресом 1.2.3.4 (шлюз)
- в локальной сети есть почтовый сервер с адресом 192.168.1.10

Для начала определим необходимые зоны: net - интернет, loc - локальная сеть, fw - CDS. В /etc/shorewall/zones поместим записи:

```
#####
#####
#ZONE TYPE OPTIONS IN OPTIONS OUT OPTIONS
#
fw firewall
net ipv4
loc ipv4
```

Теперь необходимо указать, что зона net обслуживается интерфейсом eth0, а зона loc - интерфейсом eth1. Входящий трафик на обоих интерфейсах будем пропускать через фильтры: tcpflags,routefilter,nosmurfs,logmartians. В /etc/shorewall/interfaces поместим записи:

```
#####
#####
#ZONE INTERFACE BROADCAST OPTIONS
net eth0 detect
tcpflags,nosmurfs,routefilter,logmartians
loc eth1 detect
tcpflags,nosmurfs,routefilter,logmartians
```

Опишем правила по умолчанию (политики) для трафика проходящего через шлюз: локальной сети (loc) и шлюзу (\$FW) разрешен доступ во все зоны, пакеты из сети (net) сбрасывается, к остальным применяется правило REJECT с логом. В /etc/shorewall/policy поместим записи:

```
#####
#####
#SOURCE DEST POLICY LOG LIMIT: CONNLIMIT:
LEVEL BURST MASK
#
loc all ACCEPT
$FW all ACCEPT
net all DROP
all all REJECT info
```

Добавим маскарадинг для пакетов исходящих из локальной сети в интернет. В /etc/shorewall/masq поместим записи:

```
#####
#####
#INTERFACE:DEST SOURCE ADDRESS PROTO PORT(S)
IPSEC MARK USER/
#
GROUP
eth0 192.168.1.0/24 1.2.3.4
```

Шлюз обеспечивающий выход в интернет локальной сети готов, теперь добавим несколько правил:

- разрешить ping шлюза из интернета
- разрешим соединение по ssh со шлюзом из интернета

- запретим выход в интернет с диапазона адресов 192.168.1.100-192.168.1.254
- перенаправим порты IMAP/IMAPS для доступа к почтового сервера из интернета
- перенаправим порт SMTP к почтовому серверу

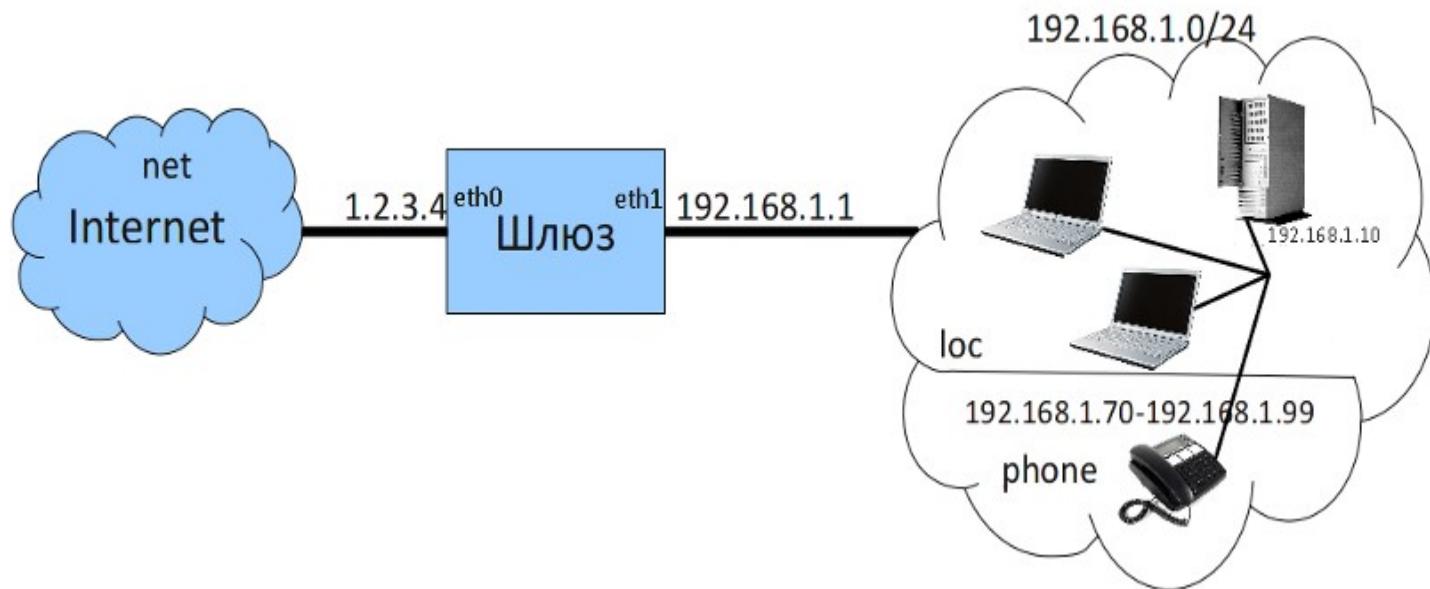
В /etc/shorewall/rules поместим записи:

```
#####
#####
#####
#ACTION SOURCE DEST PROTO DEST SOURCE
ORIGINAL RATE USER/ MARK CONNLIMIT TIME
DEST LIMIT GROUP
#SECTION ESTABLISHED
#SECTION RELATED
SECTION NEW
Ping/ACCEPT net $FW
SSH/ACCEPT net $FW
REJECT loc:192.168.1.100-192.168.1.254 net
IMAP/DNAT net loc:192.168.1.10
IMAPS/DNAT net loc:192.168.1.10
SMTP/DNAT net loc:192.168.1.10
```

Разрешаем запуск shorewall, для этого устанавливаем параметр STARTUP\_ENABLED=Yes в файле /etc/shorewall/shorewall.conf.

## Добавление ip телефонии

Модифицируем сеть предыдущего примера, добавив в нее ip телефоны:



- телефоны подключены к основной сети и их ip адреса находятся в диапазоне 192.168.1.70-192.168.1.99
- на сервер 192.168.1.10 устанавливается asterisk

Объявим новую зону phone для телефонов. Изменим /etc/shorewall/zones (зона phone должна быть объявлена обязательно перед зоной loc):

```
#####
#####
#####
###
```

```

#ZONE TYPE OPTIONS IN OPTIONS OUT OPTIONS
#
fw firewall
net ipv4
phone ipv4 # Новая строка
loc ipv4

```

Выделим телефоны из зоны loc в зону phone. Поместим в /etc/shorewall/hosts

```

#####
#####
#ZONE HOST(S) OPTIONS
phone eth1:192.168.1.70-192.168.1.99

```

Опишем политику запрещающую телефонам (phone) доступ к интернету (net). Изменим /etc/shorewall/policy

```

#####
#####
#SOURCE DEST POLICY LOG LIMIT: CONNLIMIT:
LEVEL BURST MASK
#
phone net DROP # Новая строка
loc all ACCEPT
$FW all ACCEPT
net all DROP
all all REJECT info

```

Сделаем проброс SIP и IAX портов на 192.168.1.10. Добавим к /etc/shorewall/rules

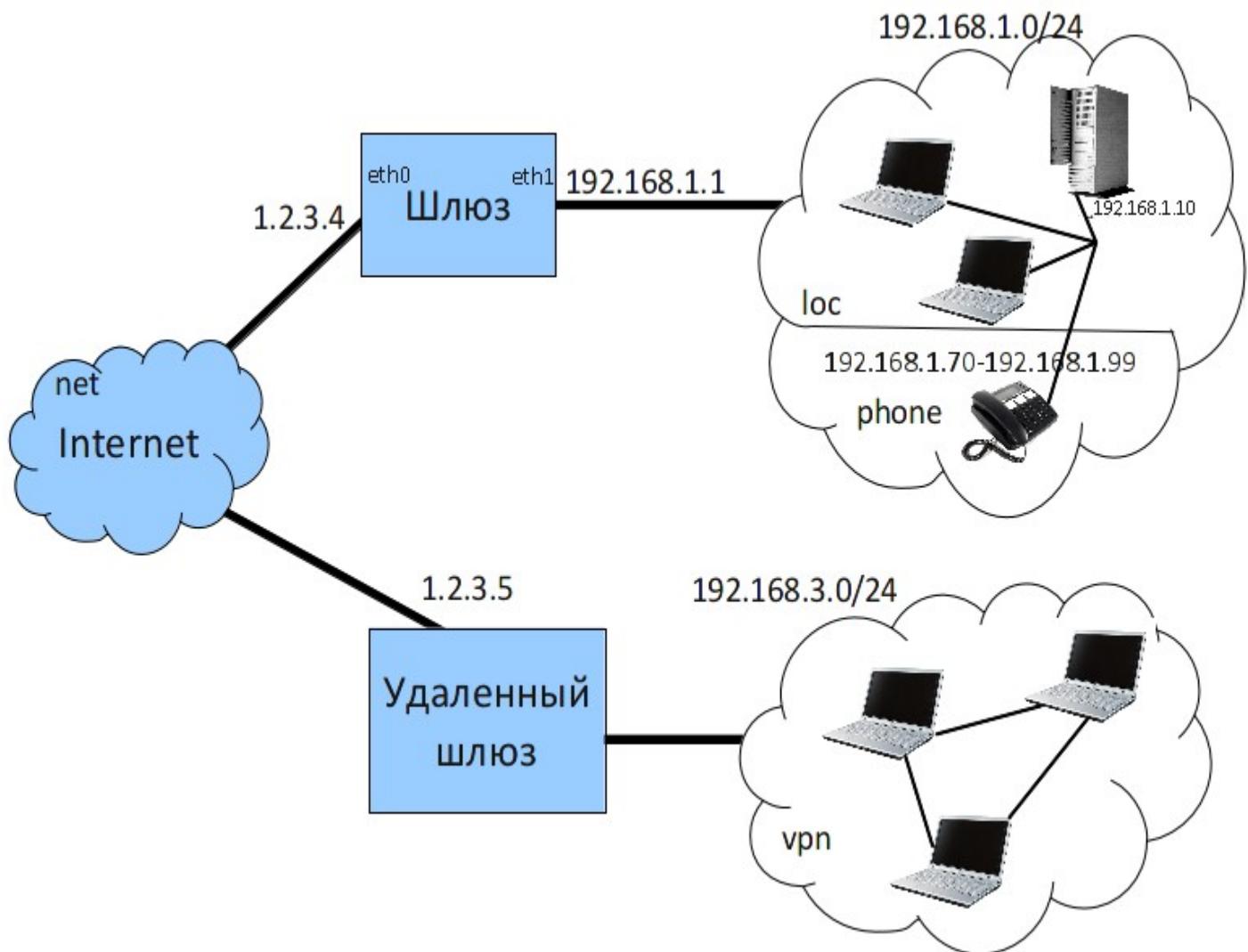
```

#####
#####
#####
#####
#ACTION SOURCE DEST PROTO DEST SOURCE
ORIGINAL RATE USER/ MARK CONNLIMIT TIME
HEADERS
#
#DEST LIMIT GROUP
#SECTION ESTABLISHED
#SECTION RELATED
SECTION NEW
...
SMTP/DNAT net loc:192.168.1.10
SIP
DNAT net loc:192.168.1.10 udp 5060
IAX
DNAT net loc:192.168.1.10 udp 4569

```

## IPSEC туннели

Модифицируем сеть предыдущего примера, в соответствии с рисунком (добавление удаленный подсети):



- Шлюз устанавливает ipsec туннель с удаленным шлюзом через интернет, подключая сегмент 192.168.3.0/24
- Удаленный шлюз имеет ip адрес 1.2.3.5

Добавляем зону `vpn` для удаленной подсети. Изменяем `/etc/shorewall/zones`.

```
#####
#####
#ZONE TYPE OPTIONS IN OPTIONS OUT OPTIONS
#
fw firewall
vpn ipv4 # Новая строка
net ipv4
phone ipv4
loc ipv4
```

Определяем зону `vpn` (пакеты, поступающие `eth0` и находящиеся в подсети 192.168.3.0/24). Изменяем `/etc/shorewall/hosts`

```
#####
#####
#ZONE HOST(S) OPTIONS
phone eth1:192.168.1.70-192.168.1.99
vpn eth0:192.168.3.0/24 # Новая строка
```

Описываем туннель: ipsec, туннель проходит через зону net, ip адрес удаленного шлюза. Записываем в /etc/shorewall/tunnels

| #TYPE | ZONE | GATEWAY | GATEWAY ZONE |
|-------|------|---------|--------------|
| ipsec | net  | 1.2.3.5 |              |

Отключаем маскарадинг пакетов отправляемых из локальной сети (loc) в удаленную подсеть (vpn). Изменяем /etc/shorewall/masq:

| #INTERFACE:DEST                                                |      |       |   |       |  | SOURCE | ADDRESS | PROTO | PORT(S) |
|----------------------------------------------------------------|------|-------|---|-------|--|--------|---------|-------|---------|
| IPSEC                                                          | MARK | USER/ | # | GROUP |  |        |         |       |         |
| eth0:192.168.3.0/24 192.168.1.0/24 1.2.3.4 # Измененная строка |      |       |   |       |  |        |         |       |         |
|                                                                |      |       |   |       |  |        |         |       |         |
|                                                                |      |       |   |       |  |        |         |       |         |

Описываем политику, разрешая соединения vpn с локальной зоной loc. Изменяем /etc/shorewall/policy:

| #SOURCE DEST |     |        |                |      |  | POLICY | LOG LEVEL | LIMIT: BURST | CONNLIMIT: MASK |
|--------------|-----|--------|----------------|------|--|--------|-----------|--------------|-----------------|
| #            |     |        |                |      |  |        |           |              |                 |
| phone        | net | DROP   |                |      |  |        |           |              |                 |
| vpn          | loc | ACCEPT | # Новая строка |      |  |        |           |              |                 |
| loc          | all | ACCEPT |                |      |  |        |           |              |                 |
| \$FW         | all | ACCEPT |                |      |  |        |           |              |                 |
| net          | all | DROP   |                |      |  |        |           |              |                 |
| all          | all | REJECT |                | info |  |        |           |              |                 |

## Пример управления трафиком

Есть канал шириной 5 мбит, необходимо разбить его следующим образом:

- SIP из интернета 256кбит гарантированно и максимально, максимальный приоритет
- SIP по туннелю 1мбит гарантированно и максимально, максимальный приоритет
- RDP по туннелю 1мбит гарантированно и весь канал минус SIP максимально (5мбит-1мбит-256кбит) обычный приоритет
- http трафик - низкий приоритет, 256кбит гарантированно и весь канал минус SIP максимально
- ftp трафик - высокий приоритет, 256кбит гарантированно и весь канал минус SIP максимально
- остальной трафик обычный приоритет, 256кбит гарантированно и весь канал минус SIP максимально

Устанавливаем ширину входящего и исходящего канала. Правим /etc/shorewall/tcdevices:

| #NUMBER:   | IN-BANDWITH | OUT-BANDWIDTH | OPTIONS | REDIRECTED INTERFACES |
|------------|-------------|---------------|---------|-----------------------|
| #INTERFACE |             |               |         |                       |
| eth0       | -           | 5mbit         |         |                       |
| eth1       | -           | 5mbit         |         |                       |

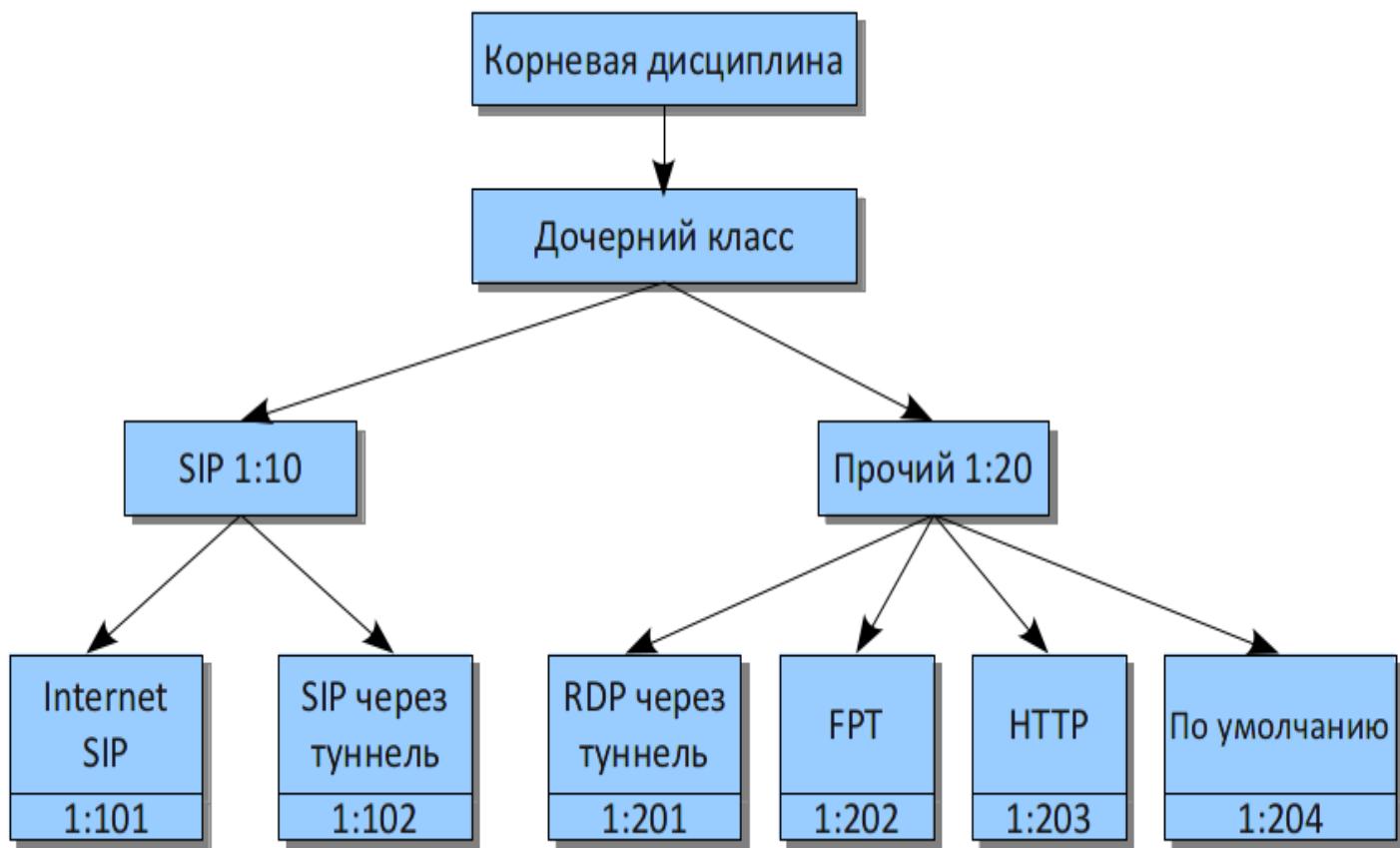
Определяем классы трафика. Так как у нас eth0 смотрит в интернет, то исходящий трафик управляетя на нем, eth1 смотрит в локальную сеть - входящий на нем. Правим /etc/shorewall/tcclasses:

| #INTERFACE:CLASS | MARK | RATE:   | CEIL     | PRIORITY |
|------------------|------|---------|----------|----------|
| #OPTIONS         |      |         |          |          |
| # DMAX:UMAX      |      |         |          |          |
| # INBOUND        |      |         |          |          |
| eth1             | 1    | 256kbit | 256kbit  | 1        |
| eth1             | 2    | 1mbit   | 1mbit    | 1        |
| eth1             | 3    | 1mbit   | 3768kbit | 3        |
| eth1             | 4    | 256kbit | 3768kbit | 2        |
| eth1             | 5    | 256kbit | 3768kbit | 4        |
| eth1             | 256  | 256kbit | 3768kbit | 3        |
| # OUTBOUND       |      |         |          |          |
| eth1             | 1    | 256kbit | 256kbit  | 1        |
| eth1             | 2    | 1mbit   | 1mbit    | 1        |
| eth1             | 3    | 1mbit   | 3768kbit | 3        |
| eth1             | 4    | 256kbit | 3768kbit | 2        |
| eth1             | 5    | 256kbit | 3768kbit | 4        |
| eth1             | 256  | 256kbit | 3768kbit | 3        |

Определяем правила, по которым трафик будет отнесен в тот или иной класс. Правим /etc/shorewall/mangle:

| #MARK                     | SOURCE         | DEST           | PROTO     | DEST       | SOURCE  | USER    |
|---------------------------|----------------|----------------|-----------|------------|---------|---------|
| TEST                      | LENGTH         | TOS            | CONNBYTES | HELPER     | PORT(S) | PORT(S) |
| #                         |                |                |           |            |         |         |
| # SIP internet            |                |                |           |            |         |         |
| MARK(1)                   | 0.0.0.0/24     | 0.0.0.0/24     | udp       | 5060       |         |         |
| MARK(1)                   | 0.0.0.0/24     | 0.0.0.0/24     | udp       | -          | 5060    |         |
| MARK(1)                   | 0.0.0.0/24     | 0.0.0.0/24     | all       | -          | -       | -       |
| -                         | -              | -              | sip       |            |         |         |
| # SIP tunnel              |                |                |           |            |         |         |
| MARK(2)                   | 192.168.3.0/24 | 192.168.1.0/24 | udp       | 5060       |         |         |
| MARK(2)                   | 192.168.1.0/24 | 192.168.3.0/24 | udp       | 5060       |         |         |
| MARK(2)                   | 192.168.3.0/24 | 192.168.1.0/24 | udp       | -          | 5060    |         |
| MARK(2)                   | 192.168.1.0/24 | 192.168.3.0/24 | udp       | -          | 5060    |         |
| MARK(2)                   | 192.168.3.0/24 | 192.168.1.0/24 | all       | -          | -       | -       |
| -                         | -              | -              | sip       |            |         |         |
| MARK(2)                   | 192.168.1.0/24 | 192.168.3.0/24 | all       | -          | -       | -       |
| -                         | -              | -              | sip       |            |         |         |
| # RDP tunnel              |                |                |           |            |         |         |
| MARK(3)                   | 192.168.3.0/24 | 192.168.1.0/24 | tcp       | 3389       |         |         |
| MARK(3)                   | 192.168.1.0/24 | 192.168.3.0/24 | tcp       | 3389       |         |         |
| MARK(3)                   | 192.168.3.0/24 | 192.168.1.0/24 | tcp       | -          | 3389    |         |
| MARK(3)                   | 192.168.1.0/24 | 192.168.3.0/24 | tcp       | -          | 3389    |         |
| # high priority FTP       |                |                |           |            |         |         |
| MARK(4)                   | 0.0.0.0/24     | 0.0.0.0/24     | all       | -          | -       | -       |
| -                         | -              | -              | ftp       |            |         |         |
| # low priority http,https |                |                |           |            |         |         |
| MARK(5)                   | 0.0.0.0/24     | 0.0.0.0/24     | tcp       | http,https |         |         |

В приведенной настройке есть особенность: так как все классы подчинены напрямую корневой дисциплине, то при отсутствии sip трафика допустим RDP и http в сумме могут занять весь канал. Если необходимо, чтобы канал выделенный под sip трафик не занимался другими трафиками - необходимо использовать иерархическую структуру подобно этой:



Для того, чтобы использовать подобную классификацию необходимо указать `classify` параметр для каждого интерфейса. Модифицируем `/etc/shorewall/tcdevices`:

```
#####
#NUMBER: IN-BANDWIDTH OUT-BANDWIDTH OPTIONS REDIRECTED
#INTERFACE
eth0 - 5mbit classify
eth1 - 5mbit classify
```

Модифицируем `/etc/shorewall/tcclasses` в соответствии со структурой представленной выше.

```
#####
#INTERFACE:CLASS MARK RATE: CEIL PRIORITY
OPTIONS
#
INBOUND
SIP
1:10 - 1256kbit 1256kbit 1
eth1:10:101 - 256kbit 256kbit 1
eth1:10:102 - 1mbit 1mbit 1

OTHER
1:20 - 3768kbit 3768kbit 2
eth1:20:201 - 1mbit 3768kbit 3
eth1:20:202 - 256kbit 3768kbit 2
```

|             |   |          |          |   |
|-------------|---|----------|----------|---|
| eth1:20:203 | - | 256kbit  | 3768kbit | 4 |
| eth1:20:204 | - | 256kbit  | 3768kbit | 3 |
| # OUTBOUND  |   |          |          |   |
| 2:30        | - | 1256kbit | 1256kbit | 1 |
| eth0:30:301 | - | 256kbit  | 256kbit  | 1 |
| eth0:30:302 | - | 1mbit    | 1mbit    | 1 |
| # OTHER     |   |          |          |   |
| 2:40        | - | 3768kbit | 3768kbit | 2 |
| eth0:40:401 | - | 1mbit    | 3768kbit | 3 |
| eth0:40:402 | - | 256kbit  | 3768kbit | 2 |
| eth0:40:403 | - | 256kbit  | 3768kbit | 4 |
| eth0:40:404 | - | 256kbit  | 3768kbit | 3 |

Определяем по новой правила, по которым трафик будет отнесен в тот или иной класс. Правим /etc/shorewall/mangle:

```
#####
#####
#MARK SOURCE DEST PROTO DEST SOURCE
USER TEST LENGTH TOS CONNBYTES PORT(S) PORT(S)
#
SIP internet
CLASSIFY(10:101) 0.0.0.0/24 0.0.0.0/24 udp 5060
CLASSIFY(10:101) 0.0.0.0/24 0.0.0.0/24 udp - 5060
CLASSIFY(10:101) 0.0.0.0/24 0.0.0.0/24 all - -
- - - - - sip
CLASSIFY(30:301) 0.0.0.0/24 0.0.0.0/24 udp 5060
CLASSIFY(30:301) 0.0.0.0/24 0.0.0.0/24 udp - 5060
CLASSIFY(30:301) 0.0.0.0/24 0.0.0.0/24 all - -
- - - - - sip
#
SIP tunnel
CLASSIFY(10:102) 192.168.3.0/24 192.168.1.0/24 udp 5060
CLASSIFY(30:302) 192.168.1.0/24 192.168.3.0/24 udp 5060
CLASSIFY(10:102) 192.168.3.0/24 192.168.1.0/24 udp - 5060
CLASSIFY(30:302) 192.168.1.0/24 192.168.3.0/24 udp - 5060
CLASSIFY(10:102) 192.168.3.0/24 192.168.1.0/24 all - -
- - - - - sip
CLASSIFY(10:302) 192.168.1.0/24 192.168.3.0/24 all - -
- - - - - sip
#
RDP tunnel
CLASSIFY(20:201) 192.168.3.0/24 192.168.1.0/24 tcp 3389
CLASSIFY(40:401) 192.168.1.0/24 192.168.3.0/24 tcp 3389
CLASSIFY(20:201) 192.168.3.0/24 192.168.1.0/24 tcp - 3389
CLASSIFY(40:401) 192.168.1.0/24 192.168.3.0/24 tcp - 3389
#
high priority FTP
CLASSIFY(20:203) 0.0.0.0/24 0.0.0.0/24 all - -
- - - - - ftp
CLASSIFY(40:403) 0.0.0.0/24 0.0.0.0/24 all - -
- - - - - ftp
#
low priority http,https
CLASSIFY(20:204) 0.0.0.0/24 0.0.0.0/24 tcp http,https
CLASSIFY(40:404) 0.0.0.0/24 0.0.0.0/24 tcp http,https
```

# Настройка Asterisk сервера

- [Настройка Asterisk сервера](#)
- [Постановка задачи](#)
- [Установка сервера Asterisk](#)
- [Конфигурация USE-флагов для устанавливаемых пакетов](#)
- [Установка пакетов](#)
- [Базовая настройка SIP](#)
- [Создание учетных записей](#)
- [Базовая настройка плана набора](#)
- [Конфигурация внутренних вызовов](#)
- [Настройка связи между двумя серверами и вызовов между ними](#)
- [Установка связи между двумя Asterisk-серверами по протоколу SIP](#)
- [Конфигурация плана набора для связи между серверами](#)
- [Настройка платы Digium AEX804E и конфигурация плана набора](#)
- [Настройка системы для работы с платой телефонии](#)
- [Объяснение понятий FXO и FXS](#)
- [Настройка FXO-каналов на Asterisk-сервере](#)
- [Конфигурация плана набора для приема вызовов](#)
- [Создание очередей вызова](#)
- [Установка своей музыки \(сообщения\) ожидания \(Music On Hold, МОН\)](#)
- [Конфигурация плана набора для совершения исходящих вызовов](#)
- [Заключение](#)

Для того чтобы настроить Calculate Directory Server в качестве IP-АТС используется Asterisk. Asterisk --- компьютерная АТС под лицензией GPL поддерживающая большое количество VoIP протоколов.

Настройка Asterisk производится путем редактирования файлов находящихся в директории /etc/asterisk.

Разберем настройку Asterisk-сервера на примере несложной конфигурации.

## Постановка задачи

- Мы имеем один настроенный Asterisk-сервер в первом офисе, нам надо настроить второй сервер для нового офиса
- На наш новый сервер мы получаем две телефонные линии по аналогу
- Одна линия используется для телефонных разговоров, вторая для факсов
- В сервере установлена плата Digium AEX804E --- PCI-E плата на 8 портов, в которую установлен один модуль на 4 FXO-порта и модуль эхоподавления
- Имеющийся сервер находится в Санкт-Петербурге, новый будет в Москве
- Сервера должны быть связаны между собой и иметь единую систему нумерации внутренних номеров
- На московском сервере должна быть следующая схема очередей звонков при поступающем вызове на основную линию:

Поступил звонок → проигрываем приветствие и предлагаем набрать добавочный номер → если добавочный номер не был набран, то переводим звонок на секретаря в СПБ → если секретарь не ответил, то переводим звонок на абонентов московского сервера → если московские абоненты не ответили, то переводим звонок на петербургских абонентов.

- При звонках поступающих на факсовую линию --- они должны адресоваться на факс
- Для подключения факса по SIP нам необходим FXS-шлюз, например, Linksys SPA2102
- Исходящие звонки должны совершаться по первой свободной линии. Если основная телефонная линия занята, то звонок идет через факсовую.
- Звонки с московского сервера в Санкт-Петербург(код 812) должны проходить через сервер в Петербурге
- Все нижеприведенные настройки тестировались и применимы для следующих версий пакетов:
  - net-misc/asterisk-1.6.2.17.3

- net-misc/dahdi-2.4.1
- net-misc/dahdi-tools-2.4.1

## Установка сервера Asterisk

Для начала работы нам необходимо установить в систему Calculate Directory Server пакет сервера Asterisk и сопутствующие пакеты.

### Конфигурация USE-флагов для устанавливаемых пакетов

Создадим файл /etc/portage/package.use/asterisk --- в нем мы пропишем все необходимые USE-флаги для нужных нам пакетов.

```
net-misc/asterisk alsa caps iconv jabber ldap samples speex ssl vorbis
dahdi span
net-misc/asterisk-core-sounds alaw g722 g729 gsm siren14 siren7 sln16 ulaw
wav
net-misc/asterisk-extra-sounds alaw g722 g729 gsm siren14 siren7 sln16
ulaw wav
net-misc/asterisk-moh-opsound alaw g722 g729 gsm siren14 siren7 sln16 ulaw
wav
media-libs/speex sse ogg
```

Для пакета net-misc/asterisk мы добавляем флаги **dahdi** (для работы с платой телефонии), **span** (для факсов) и **vorbis** (опционально для поддержки кодека vorbis). У остальных пакетов мы прописываем поддержку всех доступных кодеков.

### Установка пакетов

Теперь, когда мы установили необходимые USE-флаги, приступим к установке.

Вводим команду:

```
emerge -a asterisk dahdi dahdi-tools
```

Мы получим такой список пакетов для установки:

These are the packages that would be merged, in order:

```
Calculating dependencies... done!
[ebuild N] net-libs/libpri-1.4.11.4
[ebuild N] sys-libs/slang-2.2.2 USE="pcre png readline zlib -cjk"
[ebuild N] dev-libs/iksemel-1.3 USE="-gnutls"
[ebuild N] media-libs/spandsp-0.0.6_pre12-r1 USE="mmx sse sse2 sse3
-doc (-fixed-point) -static-libs
[ebuild N] net-misc/dahdi-2.4.1 USE="flash"
[ebuild N] dev-libs/newt-0.52.12 USE="gpm nls -tcl"
[ebuild N] net-misc/dahdi-tools-2.4.1 USE="-ppp"
[ebuild N] net-misc/asterisk-1.6.2.17.3 USE="alsa caps dahdi iconv
jabber ldap samples span speex ssl vorbis -doc -freetds -lua -newt -oss
-postgres -radius -snmp -sqlite"
[ebuild N] net-misc/asterisk-extra-sounds-1.4.11 USE="alaw g722
g729 gsm siren14 siren7 sln16 ulaw wav" LINGUAS="fr"
[ebuild N] net-misc/asterisk-moh-opsound-2.03 USE="alaw g722 g729
gsm siren14 siren7 sln16 ulaw wav"
[ebuild N] net-misc/asterisk-core-sounds-1.4.19 USE="alaw g722 g729 gsm siren14 siren7 sln16 ulaw wav"
LINGUAS="fr"
```

Would you like to merge these packages? [Yes/No]

Соглашаемся на установку пакетов и ждем. После того, как все будет установлено, можно будет приступить к настройке.

## Базовая настройка SIP

Задача поставлена --- приступим к настройке нашего нового сервера.

### Создание учетных записей

Сперва добавим несколько локальных SIP-абонентов. Для этого нам необходимо редактировать файл /etc/asterisk/sip.conf

Сначала добавим шаблон для общих настроек sip-телефонов:

```
[office-phones](!)
type=friend
context=outcoming-sip
secret=xxxxxxxx
host=dynamic
nat=no
qualify=yes
canreinvite=no
callgroup=1
pickupgroup=1
dtmfmode=auto
disallow=all
allow=g722
```

Рассмотрим этот участок кода. Шаблон задается при помощи восклицательного знака в скобках около имени секции [phones](!)

- type --- тип соединения (peer --- только исходящие вызовы, user --- входящие вызовы, friend --- входящие и исходящие вызовы)
- context --- контекст в /etc/asterisk/extension.conf в который будет выполняться при звонке с телефона
- secret --- пароль для SIP телефона
- host --- имя хоста телефона (dynamic динамическое имя хоста)
- nat --- сетевая трансляция адресов
- qualify --- проверка функционирования телефона
- canreinvite --- прямой доступ телефона к другому телефону минуя Asterisk
- callgroup --- номер телефонной группы
- pickupgroup --- номер группы захвата звонков
- dtmfmode=auto --- dtmf режим (auto | inband | info | rfc2833)
- disallow=all --- запрещает все кодеки
- allow=g722 --- разрешает использование кодека g722

Теперь добавим секции для каждого телефона:

```
[001](office-phones)
callerid="Менеджер 1" <800>
[002](office-phones)
callerid="Менеджер 2" <801>
[003](office-phones)
t38pt_udptl=yes
callerid="Факс МСК" <802>
disallow=all
allow=alaw
allow=ulaw
```

Для факса мы переопределяем используемые кодеки, т. к. факсовый сигнал нормально проходит лишь при использовании кодека g711(alaw и ulaw) и t38pt\_udptl=yes разрешает использование протокола t.38 для передачи факсов.

## Базовая настройка плана набора

Наши телефоны и FXS-шлюз теперь могут зарегистрироваться на сервере, но звонить пока они еще не могут. Для этого нам надо создать контекст [outcoming-sip] в файле /etc/asterisk/extensions.conf

## Конфигурация внутренних вызовов

Для удобства выделим локальные номера в отдельную секцию:

```
[msk-local-phones]
exten => 800,1,Log(NOTICE,"800 ACCOUNT")
exten => 800,2,Dial(SIP/001,120,Tt)
exten => 801,1,Log(NOTICE,"801 ACCOUNT")
exten => 801,2,Dial(SIP/002,120,Tt)
exten => 802,1,Log(NOTICE,"802 ACCOUNT")
exten => 802,2,Dial(SIP/003,120,Tt)
```

И сделаем секцию завершающую вызов:

```
[handup-sip]
exten => _X!,1,HangUp()
```

В секцию [outcoming-sip] теперь подключим наши секции:

```
[outcoming-sip]
include => msk-local-phones
include => handup-sip
```

Теперь перезагрузив настройки Asterisk мы сможем звонить по номерам внутри сервера 800, 801 и 802 соответственно. Перезагрузка настроек asterisk выполняется следующим образом. Запускаем консоль управления Asterisk-сервером командой asterisk -r и прописываем в консоли reload --- это заставит Asterisk перечитать все настройки, либо же можно перечитать их по отдельности прописав sip reload для перезагрузки настроек sip и dialplan reload для перезагрузки настроек плана набора. Введя команду sip show peers мы можем посмотреть, какие абоненты у нас могут присутствовать на сервере и кто сейчас зарегистрирован и с какого IP.

## Настройка связи между двумя серверами и вызовов между ними

Наши абоненты уже звонят друг-другу, но у нас есть еще и абоненты на нашем сервере в Петербурге, нам надо чтобы они могли звонить и им тоже. Для этого нам надо для начала настроить связь между нашими двумя серверами Asterisk.

## Установка связи между двумя Asterisk-серверами по протоколу SIP

На сервере в Москве прописываем в файле /etc/asterisk/sip.conf следующее:

В секции [general]:

```
register => msk_asterisk:XXXXXXXXXX@192.168.0.30/spb_asterisk
```

И в конце создаем новую секцию:

```
[spb_asterisk]
type=friend
secret=XXXXXXXXXX
context=spb_incoming
host=dynamic
```

```
qualify=yes
dtmfmode=rfc2833
disallow=all
allow=ulaw
```

На сервере в Санкт-Петербурге пишем очень похожим образом:

В секции [general]:

```
register => spb_asterisk:XXXXXXXXXX@192.168.1.25/msk_asterisk
```

И в конце создаем новую секцию:

```
[msk_asterisk]
type=friend
secret=XXXXXXXXXX
context=msk_incoming
host=dynamic
qualify=yes
dtmfmode=rfc2833
disallow=all
allow=ulaw
```

Теперь рассмотрим подробнее что-же мы такое написали и для чего.

```
register => msk_asterisk:XXXXXXXXXX@192.168.0.30/spb_asterisk
```

Это указывает нашему московскому серверу зарегистрироваться на петербургском сервере **192.168.0.30/spb\_asterisk** с логином **msk\_asterisk** и паролем **XXXXXXXXXX**.

Эти логин и пароль мы как раз задали, создав секцию **[msk\_asterisk]** на нашем сервере в Петербурге. И соответственно петербургский сервер регистрируется на московском **192.168.1.25/msk\_asterisk** с логином и паролем указанными секцией **[spb\_asterisk]** в конфигурации московского сервера.

## Конфигурация плана набора для связи между серверами

Связь между серверами установлена --- необходимо описать план набора для того чтобы московские абоненты могли звонить петербургским и наоборот.

В файле /etc/asterisk/extensions.conf на московском сервере создаем секцию **[spb-local-phones]**, в который мы пропишем наших петербургских абонентов:

```
[spb-local-phones]
exten => 500,1,Dial(SIP/spb_asterisk/500,120,Tt)
exten => 501,1,Dial(SIP/spb_asterisk/501,120,Tt)
(... и т. д.)
```

Отметим небольшую разницу с тем, как мы прописывали локальных абонентов.

Если локальным абонентам мы звонили **exten => 800,2,Dial(SIP/001,120,Tt)**, то есть набрав номер 800 мы совершили звонок абоненту, который зарегистрирован с логином 001, то в данном случае **exten => 500,1,Dial(SIP/spb\_asterisk/500,120,Tt)** мы звоним абоненту находящемуся на сервере spb\_asterisk с телефонным номером 500.

В имеющуюся секцию **[outcoming-sip]** добавляем включение новой секции с локальными номерами петербургского сервера:

```
[outcoming-sip]
include => msk-local-phones
include => spb-local-phones
```

```
include => handup-sip
```

И создаем секцию [spb\_incoming] для обработки вызовов поступающих с петербургского сервера:

```
[spb_incoming]
include => msk-local-phones
include => handup-sip
```

На сервере в Петербурге прописываем аналогичным образом:

```
[msk-local-phones]
exten => 800,1,Dial(SIP/msk_asterisk/800,120,Tt)
exten => 801,1,Dial(SIP/msk_asterisk/801,120,Tt)
exten => 802,1,Dial(SIP/msk_asterisk/802,120,Tt)
```

```
[outcoming-sip]
include => msk-local-phones
include => spb-local-phones
include => handup-sip
```

```
[msk_incoming]
include => spb-local-phones
include => handup-sip
```

Перегружаем настройки SIP и плана набора. Теперь наши абоненты в Москве и Петербурге (на обоих серверах установлена система Calculate Directory Server с пакетом Asterisk) могут общаться между собой.

## Настройка платы Digium AEX804E и конфигурация плана набора

Наши абоненты звонят друг-другу, звонят абонентам петербургского сервера, но они пока еще не могут звонить на городские телефоны. Приступим к настройке нашей платы телефонии.

### Настройка системы для работы с платой телефонии

Для работы платы телефонии необходимы пакеты net-misc/dahdi и net-misc/dahdi-tools.

Когда они установлены, то закомментируем строку с модулем для нашей платы в файле /etc/modprobe.d/dahdi.blacklist.conf для того чтобы модуль для нее загружался системой.

Файл примет следующий вид:

```
blacklist all the drivers by default in order to ensure that
/etc/init.d/dahdi installs them in the correct order so that the spans
are
ordered consistently.
```

```
blacklist wct4xxp
blacklist wcte12xp
blacklist wct1xxp
blacklist wcte11xp
#blacklist wctdm24xxp
blacklist wcfxo
blacklist wctdm
blacklist wctc4xxp
blacklist wcb4xxp
blacklist wcopenpci
blacklist zaphfc
```

В нашем случае нам необходим модуль `wctdm24xxxr` --- этот модуль используется платами TDM2400P/AEX2400, TDM800P/AEX800 и TDM410P/AEX410.

Теперь настроим каналы на нашей плате. Для этого нам необходимо отредактировать файл `/etc/dahdi/system.conf`

В нашей плате установлен модуль на 4 FXO канала, с номерами каналов от 5 до 8(если смотреть на плату, то это 4 левых гнезда. На этой плате нумерация каналов идет справа налево. Восьмой канал --- крайний левый, первый канал --- крайний правый.

В конфигурационном файле прописываем следующее:

```
fxsks=5
fxsks=6
fxsks=7
fxsks=8
```

Это указывает нам, что для каналов с 5-го по 8-й у нас используется сигнализация `fxsks`.

## Объяснение понятий FXO и FXS

Немного отвлечемся и разберемся, что есть FXO и FXS и где какая сигнализация.

Порт FXO принимает телефонную линию с телефонной станции, то есть ведет себя как оконечное оборудование(телефонный аппарат) и использует сигнализацию `fxsks`.

Порт FXS генерирует сигнал готовности линии и выдает напряжение линии, то есть ведет себя как телефонная станция и использует сигнализацию `fxoks`.

Таким образом порты называются исходя из того, что к ним подключается и используют сигнализацию исходя из того, чем они являются сами.

## Настройка FXO-каналов на Asterisk-сервере

Теперь мы запускаем демон `dahdi` `/etc/init.d/dahdi start` и смотрим, что у нас все работает как надо командой `dahdi_cfg -vv`

Вывод команды:

```
DAHDI Tools Version - 2.4.1
```

```
DAHDI Version: 2.4.1
```

```
Echo Canceller(s):
```

```
Configuration
```

```
=====
```

Channel map:

```
Channel 05: FXS Kewlstart (Default) (Echo Canceler: none) (Slaves: 05)
Channel 06: FXS Kewlstart (Default) (Echo Canceler: none) (Slaves: 06)
Channel 07: FXS Kewlstart (Default) (Echo Canceler: none) (Slaves: 07)
Channel 08: FXS Kewlstart (Default) (Echo Canceler: none) (Slaves: 08)
```

```
4 channels to configure.
```

```
etting echocan for channel 5 to none
```

```
Setting echocan for channel 6 to none
```

```
Setting echocan for channel 7 to none
```

```
Setting echocan for channel 8 to none
```

Теперь необходимо настроить каналы в самом Asterisk-сервере. Для этого мы редактируем файл `/etc/asterisk/chan_dahdi.conf`

В нем нам необходимо отредактировать секцию [channels]:

```
[channels]
usecallerid=yes
hidecallerid=no
callwaiting=no
threewaycalling=yes
transfer=yes
echocancel=yes
echotraining=yes
signaling=fxs_ks
context=incoming_fxo
group=1
channel=>7
context=incoming_fxo_fax
group=1
channel=>8
```

Рассмотрим параметры:

- usecallerid=yes --- активирует возможность передачи CallerID
- hidecallerid=no --- указывает, что для исходящих вызовов будет передаваться CallerID
- callwaiting=no --- деактивирует отложенный вызов
- threewaycalling=yes --- активирует возможность подключения третьего абонента
- transfer=yes --- разрешает переадресацию вызова
- echocancel=yes --- активирует эхоподавление
- echotraining=yes --- активирует режим измерения эха для ускорения настройки эхоподавления
- signaling=fxs\_ks --- указывает на использование сигнализации fxs
- group=1 --- группа каналов

Первый контекст указывает как будет обрабатываться вызов для 7-го канала, второй для 8-го. Седьмой канал мы используем для основной линии, восьмой --- для факсов. Параметр group=1 для обоих каналов объединяет их в одну группу. Это позволит нам совершать вызовы через первую свободную линию.

## Конфигурация плана набора для приема вызовов

Настроим прием вызовов на наши линии.

Создаем контекст [incoming\_fxo] в файле /etc/asterisk/extensions.conf:

```
[incoming_fxo]
exten => s,1,Answer()
exten => s,n,Ringing()
exten => s,n,Queue(welcome,n,,,12)
exten => s,n,GotoIfTime(19:15-8:00, *, *, *?allRing:default)
exten => s,n(allRing),NoOp()
exten => s,n,Queue(allNoFaxes,r,,,600)
exten => s,n,HangUp()
exten => s,n(default),NoOp()
exten => s,n,Queue(secretary,r,,,6)
exten => s,n,Queue(msk-manager1,r,,,10)
exten => s,n,Queue(spb-managers,r,,,600)
include => handup-sip
```

Здесь мы отвечаем на вызов и отправляем его в необходимую очередь. Queue(welcome, n, , , 12) отправляет звонок в очередь welcome на 12 секунд, параметр «n» позволяет набор во время нахождения в очереди и проигрывается музыка ожидания(Music On Hold, МОН). В очереди welcome мы проигрываем приветственное сообщение --- это будет описано ниже. GotoIfTime(19:15-8:00, \*, \*, \*?allRing:default) --- проверяется условие, если сейчас время от 19:15 вечера до 8:00 утра, то мы переходим к метке allRing, иначе к метке default. Это делается для того, чтобы в то время, когда большинства работников нет в офисе звонили все телефоны, а в рабочие часы --- в установленном порядке. Далее под меткой default у нас звонок сначала на 6 секунд идет на секретаря, затем на 10 секунд на московских менеджеров и потом на 600 секунд на петербургских менеджеров. Параметр «g» в вызове Queue() отвечает за то, что вместо МОН звонящий слышит гудки, как при звонке (ringing).

Также нам нужен контекст для второй линии --- создадим секцию [incoming\_fxo\_fax] в файле /etc/asterisk/extensions.conf:

```
[incoming_fxo_fax]
exten => s,1,Answer()
exten => s,2,NoOp()
exten => s,n,Ringing()
exten => s,n,Queue(msk-faxes,r,,600)
include => handup-sip
```

На этой линии мы просто отправляем поступивший звонок на факс на 600 секунд.

## Создание очередей вызова

Теперь рассмотрим, как именно описываются очереди. Они описываются в файле /etc/asterisk/queues.conf.

```
[welcome]
strategy = ringall
musicclass=welcome
context=all-local-sip

[secretary]
strategy = ringall
musicclass=default
context=all-local-sip
; Секретарь
member => SIP/spb_asterisk/550,1

[spb-managers]
strategy = ringall
musicclass=default
context=all-local-sip
; Менеджеры
member => SIP/spb_asterisk/500,1
member => SIP/spb_asterisk/501,1
; (... и т. д.)

[msk-faxes]
strategy = ringall
musicclass=default
context=all-local-sip
; Московский факс
member => SIP/002,1
```

```
[msk-manager1]
strategy = ringall
musicclass=default
context=all-local-sip
; Московские менеджеры
member => SIP/001,1
```

Основные параметры при описании очередей:

- strategy --- стратегия по которой обрабатываются поступающие вызовы, может принимать несколько значений:
  - ringall --- вызываются все доступные участники до тех пор, пока кто-то из них не ответит на вызов (по умолчанию)
  - leastrecent --- вызывается участник очереди, который вызывался меньше всего
  - fewestcalls --- вызывается участник очереди, ответивший на наименьшее количество звонков
  - random --- случайным образом из участников очереди вызывается свободный
  - rrmemory --- циклический вызов с памятью, запоминается последний ответивший участник очереди
- musicclass --- задает, какая музыка или сообщение проигрывается во время ожидания в данной очереди(задается в файле /etc/asterisk/musiconhold.conf)
- context --- задает, какой контекст будет обрабатывать набор номера во время ожидания.
- member --- задает участников очереди, обрабатывающих вызовы. На каждого участника создается запись.

Для обработки добавочного номера в очереди welcome мы указали контекст [all-local-sip] --- он должен включать в себя все локальные московские и петербургские номера, создадим для него запись в файле /etc/asterisk/extensions.conf

```
[all-local-sip]
include => msk-local-sip
include => spb-local-sip
include => handup-sip
```

Постановка звонка в очередь и обработка происходит следующим образом:

Функцией Queue(welcome, n, , , 12) в плане набора мы отправляемый поступивший звонок на обработку в очередь welcome. В ней в течении 12 секунд звонящий слышит приветственное сообщение и, так как мы передали параметр "n", может набрать добавочный номер, который обрабатывается контекстом all-local-sip. По истечении 12 секунд, если не был набран добавочный номер, звонок выходит из данной очереди и идет на следующий шаг обработки в плане набора.

При вызове функции Queue(msk-manager1, r, , , 10) звонок переходит на обработку в очередь msk-manager1 на 10 секунд. При нахождении в очереди, так как мы указали параметр "r", звонящий слышит гудки. В это время звонок обрабатывается в соответствии с заданной стратегией --- в данном случае ringall (звонят телефоны у всех указанных членов очереди). Если кто-то из членов очереди берет трубку, то устанавливается соединение. В противном случае, если за 10 секунд трубка взята не была, то обработка вызова снова переходит к следующему шагу в плане набора.

## Установка своей музыки (сообщения) ожидания (Music On Hold, МОН)

У нас уже обрабатываются входящие на наши городские линии и мы проигрываем приветственное сообщение, указав для очереди welcome музыкальный класс welcome --- рассмотрим, как именно он задается. Музыка ожидания описывается в файле /etc/asterisk/musiconhold.conf

При установке asterisk он имеет следующий вид:

```
[general]
```

```
[default]
mode=files
directory=moh
```

Нам надо добавить в него следующую секцию:

```
[welcome]
mode=files
sort=alpha
directory=/etc/asterisk/moh1
```

В данной записи мы говорим Asterisk-серверу проигрывать файлы в алфавитном порядке из директории /etc/asterisk/moh1. В эту директорию мы помещаем записанное сообщение. Для уменьшения нагрузки на сервере предварительно сконвертируем при помощи ffmpeg (пакет media-video/ffmpeg предустановлен в системе Calculate Linux Desktop) наше сообщение в наиболее часто используемые кодеки alaw, ulaw, g722, g729, gsm и в Asterisk Native SLN из которого Asterisk сможет конвертировать сам налету.

Например конвертация в Native SLN при помощи ffmpeg делается следующим образом:

```
ffmpeg -i "[input file]" -ar 8000 -ac 1 -acodec pcm_s16le -f s16le
"[output file].sln"
```

Сконвертировать имеющиеся mp3 файлы в mono wav и ulaw pcm можно следующим образом:

```
for f in `ls *.mp3` ; do FILE=$(basename $f .mp3) ; ffmpeg -i $FILE.mp3
-ar 8000 -ac 1 -ab 64 $FILE.wav -ar 8000 -ac 1 -ab 64 -f mulaw $FILEpcm
-map 0:0 -map 0:0 ; done
```

## Конфигурация плана набора для совершения исходящих вызовов

Теперь нам осталось настроить исходящие вызовы на городские номера. В файле /etc/asterisk/extensions.conf создаем секцию [city-calls]:

```
[city-calls]
exten => _98812XXXXXXX,1,Dial(SIP/spb_asterisk/${EXTEN})
exten => _98812XXXXXXX,2,Congestion
exten => _9XXXXXXX,1,Dial(DAHDI/g1/8495${EXTEN:1})
exten => _9XXXXXXX,2,Congestion
exten => _98.,1,Dial(DAHDI/g1/${EXTEN:1})
exten => _98.,2,Congestion
```

И подключаем ее к секции [outcoming-sip]:

```
[outcoming-sip]
include => local-sip
include => spb-local-sip
include => city-calls
include => handup-sip
```

Таким образом вызовы на петербургские номера (код 812) у нас отправляются на сервер spb\_asterisk и обрабатываются уже там. Вызовы на городские номера (7 цифр) дополняются до 10 цифр с кодом города (в данном случае в начало номера добавляется 8495 --- сервер у нас находится в Москве) и вызовы на межгород вызываются как есть.

В вызове команды Dial(DAHDI/g1/\${EXTEN:1}):

- DAHDI --- указывает на то, что мы совершаем звонок через нашу FXO-плату

- g1 --- группа каналов 1 (мы это задавали при конфигурации FXO-каналов), звонок совершается через группу для того чтобы, если основная линия занят он шел через факсовую линию
- \${EXTEN} и \${EXTEN:1} --- номер, который был набран. Полный синтаксис команды \${EXTEN:x:y}, где x --- начальное положение и y --- количество возвращаемых цифр.

## Заключение

Таким образом мы настроили Asterisk-сервер в базовой конфигурации, соответствующей поставленной задаче. Это лишь основные настройки, позволяющие совершать и принимать необходимые вызовы. Сервер телефонии Asterisk предоставляет неограниченные возможности по настройке и конфигурации офисной IP-АТС. Надеемся, что данная статья поможет вам разобраться с настройкой Asterisk и послужит фундаментом, основываясь на котором, вы сможете создавать свою собственную конфигурацию.

# Обзор структуры LDAP сервера

## Введение

Записи LDAP сервера состоят из одного или нескольких атрибутов и обладают уникальным именем (DN --- *Distinguished Name*). Уникальное имя может выглядеть, например, следующим образом: "cn=Иван Петров,ou=Сотрудники,dc=example,dc=com".

Уникальное имя состоит из одного или нескольких *относительных уникальных имен* (RDN --- *Relative Distinguished Name*), разделённых запятой. Относительное уникальное имя имеет вид "*ИмяАтрибута=значение*". На одном уровне каталога не может существовать двух записей с одинаковыми относительными уникальными именами. В силу такой структуры уникального имени записи в каталоге можно легко представить в виде дерева, где записи будут ветвями, а в роли листьев будут выступать атрибуты данных записей. В качестве корня дерева (*корневой ветки*) выступает запись которой подчинены другие записи, а сама она никому не подчинена.

## Общее описание структуры LDAP Calculate сервера

Корневая ветка "dc=calculate" содержит следующие записи ("ветки"):

`cn=ldapadmin`

содержит DN и пароль для подключения к LDAP серверу с полными правами.

`cn=proxyuser`

содержит DN и пароль подключения к LDAP серверу с правами только для чтения, так же недоступны будут такие поля как userPassword и т.д.

`ou=Services`

содержит в себе ветки сервисов, установленные посредством утилит *Calculate 2*.

Таким образом ветка "ou=Services" содержит набор ветвей описания сервисов `ou=Unix`, `ou=Mail`, `ou=Samba`, `ou=Ftp` и `ou=Jabber`. Каждая ветка сервиса является DN и может хранить пароль подключения к LDAP с доступом к их данным и может содержать дочерние ветки: "ou=Users" (учетные записи), "ou=Groups" (группы), "ou=Computers" (компьютеры).

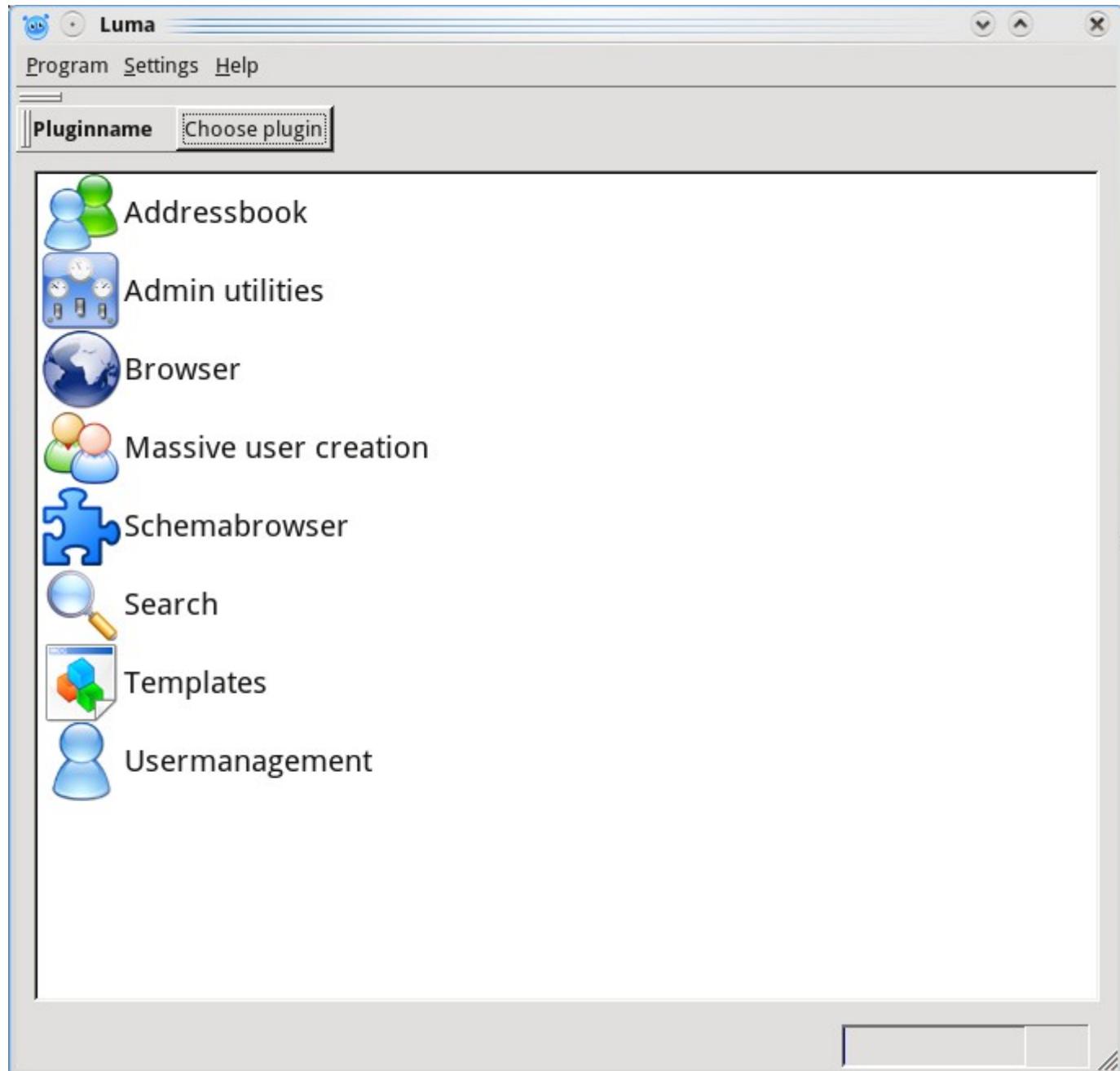
## Просмотр структуры LDAP

Для просмотра структуры LDAP сервера вы можете воспользоваться любым LDAP браузером.

Для подключения к базе с полными правами требуется пароль администратора LDAP базы. Узнать его можно в файле `/etc/calculate/calculate.ldap` на сервере. Файл разбит на области, названия которых соответствуют названиям сервисов. В каждой секции находится запись DN и пароль для подключения. Учетная запись администратора находится в секции `[admin]`.

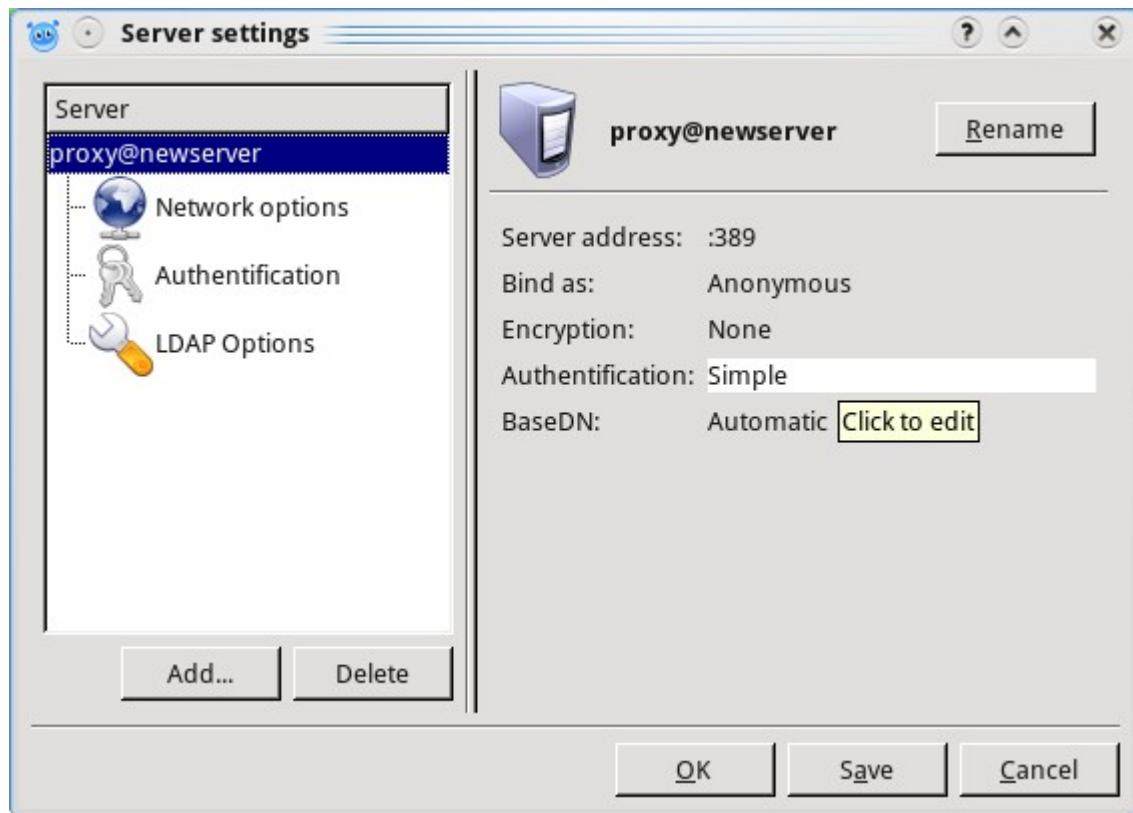
## Просмотр структуры LDAP на примере программы Luma

После запуска программы Luma откроется окно:



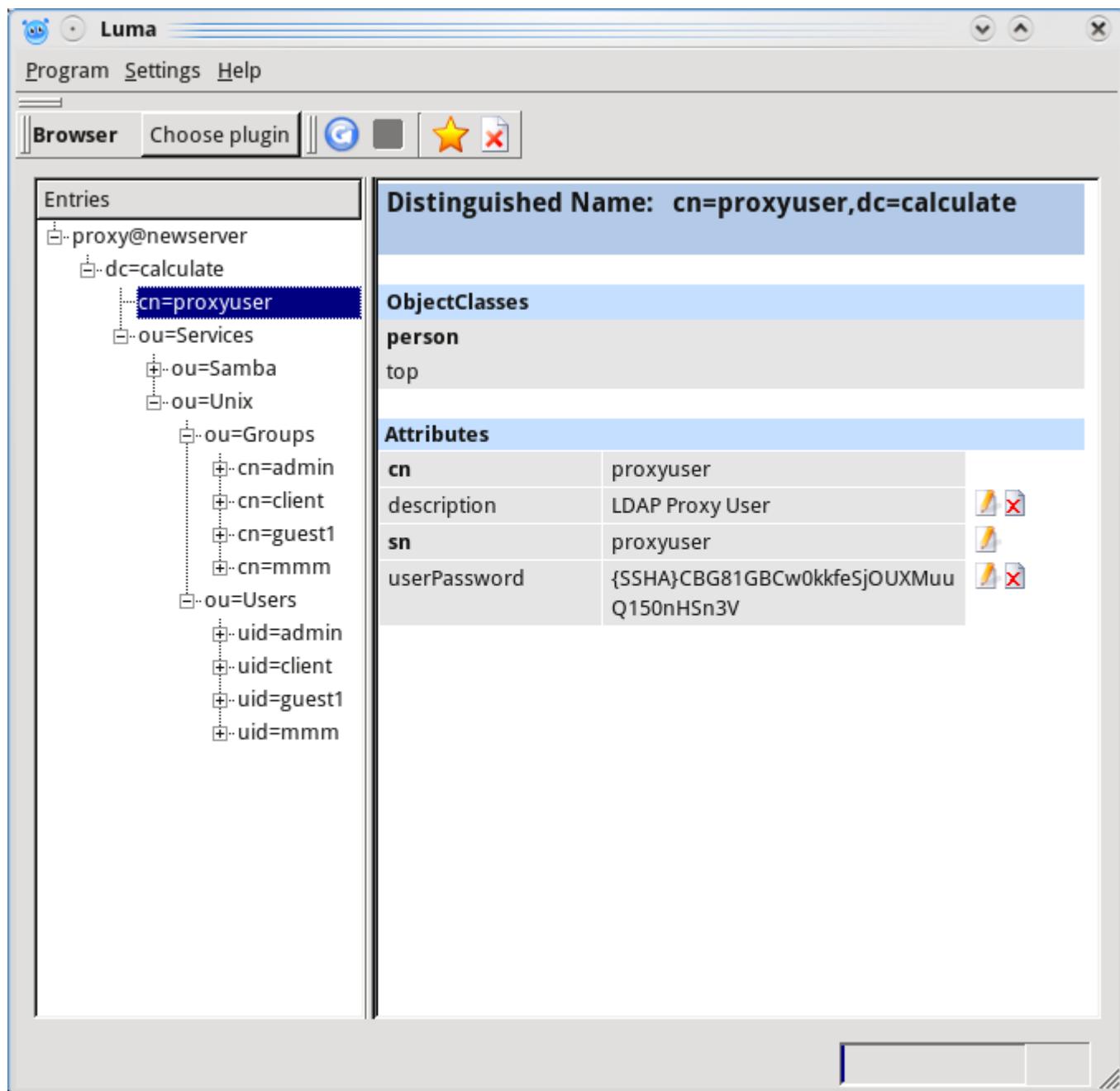
Создадим подключение к LDAP базе сервера с полным доступом, для этого

- Выбираем пункт меню *Settings->Edit Server List* или нажимем *Ctrl+E*, после чего откроется окно для управления подключениями к LDAP серверам:



- нажимаем кнопку *Add* и в появившееся поле для ввода вписываем произвольное название подключения, например *proxy@newserver*
- нажимаем на появившееся название в списке *Server*, и откроются ветки настройки подключения к LDAP серверу
- щелкаем ветку *Network options* и в появившихся настройках справа вписываем *Hostname* сервера
- переходим к ветке *Authentification* и снимаем пометку *Anonymous bind*
- оставляем *Mechanism* со значением *Simple*
- вписываем в *Bind as* значение *DN - "cn=ldapadmin,dc=calculate"*, а в *Password* содержимое поля *PASS*, файла */etc/calculate/calculate.ldap*, нажимаем *OK*

Выбираем плагин *Browser* и в открывшемся окне смотрим структуру LDAP сервера:



## Управление клиентскими машинами

### Введение

Одной из основных задач дистрибутивов Calculate Linux - упростить задачи администрирования большого количества систем. В настоящем руководстве речь пойдет о клиентских машинах - [Calculate Linux Desktop](#), введенных в домен [CDS](#).

### Сетевой диск «remote»

Основные ресурсы системы вынесены в директорию `/var/calculate/remote`. Здесь можно увидеть директории:

- `assemble` - дополнительные ресурсы для сборки системы
- `distfiles` - исходники программ
- `linux` - дистрибутивы
- `packages` - бинарные пакеты программ
- `snapshots` - архивы портейжей
- `stages` - `stage3` образы для компиляции системы

- `templates` - пользовательские шаблоны

При [вводе Calculate Linux Desktop в домен](#) Calculate Directory Server, директория `/var/calculate/remote` монтируется при помощи сетевой файловой системы `cifs` с сервера. Обратите внимание, директория монтируется пользователем "client".

## Шаблоны установки

Все дистрибутивы Calculate Linux для настройки системы используют [шаблоны](#). Это достаточно простой и эффективный прием для изменения настроек системы под свои нужды. Пользовательские шаблоны расположены в директории `/var/calculate/remote/templates`. Используя общий ресурс, вы можете менять настройки на всех машинах, либо выборочно, применяя в названии шаблона например сетевое имя машины, либо для всех машин.

## Выполнение команд на стороне клиента

Может возникнуть ситуация, когда вам понадобится выполнить определенные действия на всех машинах локальной сети. Например вы можете захотеть узнать кто работает за системой, как долго работает каждая машина без перезагрузки (`uptime`), сколько пользователей одновременно зашли в сеанс, не было ли ошибок при инициализации графической сессии и т.д. Для этого вы можете использовать открытый ключ, сгенерированный на сервере и размещенный на клиентских машинах в директории `/root`. В этом случае вы получаете полный доступ из скрипта на машины клиентов.

Создание открытого ключа описано в статье [Резервное копирование](#). Не забывайте о шаблонах, при помощи которых можно записывать открытый ключ на компьютеры пользователей.

Вы можете написать скрипт сканирования сети, выявляющий работающие Linux клиенты. Попробуйте использовать следующую команду для поиска:

```
LANG=C nmap -p111 -n -PS111 192.168.0.0/24
```

Выполнить удаленно команду на клиенте можно при помощи следующей команды:

```
ssh -o 'StrictHostKeyChecking=no' root@HOST EXEC
```

Где HOST - сетевое имя либо IP адрес машины, EXEC - команда.

## Обновления из бинарных пакетов

У менеджера пакетов [emerge](#) есть замечательная возможность создавать бинарные архивы собираемых программ и устанавливать из них пакеты. Что радует, для установки бинарных программ вовсе не обязательно иметь определенную версию портежей. Фактически вы можете обновлять портежи только на одном клиенте, предоставив ему доступ к записи в директорию `/var/calculate/remote/packages`.

Для разделения пакетов собираемых в разных системах с разными архитектурами, Calculate Linux модифицирует переменную `PKGDIR` в профиле дистрибутива.

Для сборки бинарного пакета выполните:

```
emerge --buildpkg PACKAGE
```

где PACKAGE имя пакета.

Установить пакет, можно указав его полное имя с версией, используя флаг "`--usepkgonly`". Обновлять портежи на каждой клиентской машине при этом вовсе не обязательно.

Удобно использовать оба параметра в одной команде:

```
emerge -bk PACKAGE
```

В этом случае пакет будет скомпилирован только при отсутствии бинарного пакета.

# Система виртуализации QEMU

- [Система виртуализации QEMU](#)
- [Установка пакетов](#)
- [Настроим параметры сети](#)
- [Создание виртуальной машины](#)
- [Virtio --- что это и с чем это едят](#)
- [Команды изменение привода](#)
- [Настройка сети](#)

**KVM** (или **Kernel-based Virtual Machine**) --- это программное решение, обеспечивающее виртуализацию в среде Linux, которая поддерживает аппаратную виртуализацию на базе **Intel VT** (**Virtualization Technology**) либо **AMD SVM** (**Secure Virtual Machine**)

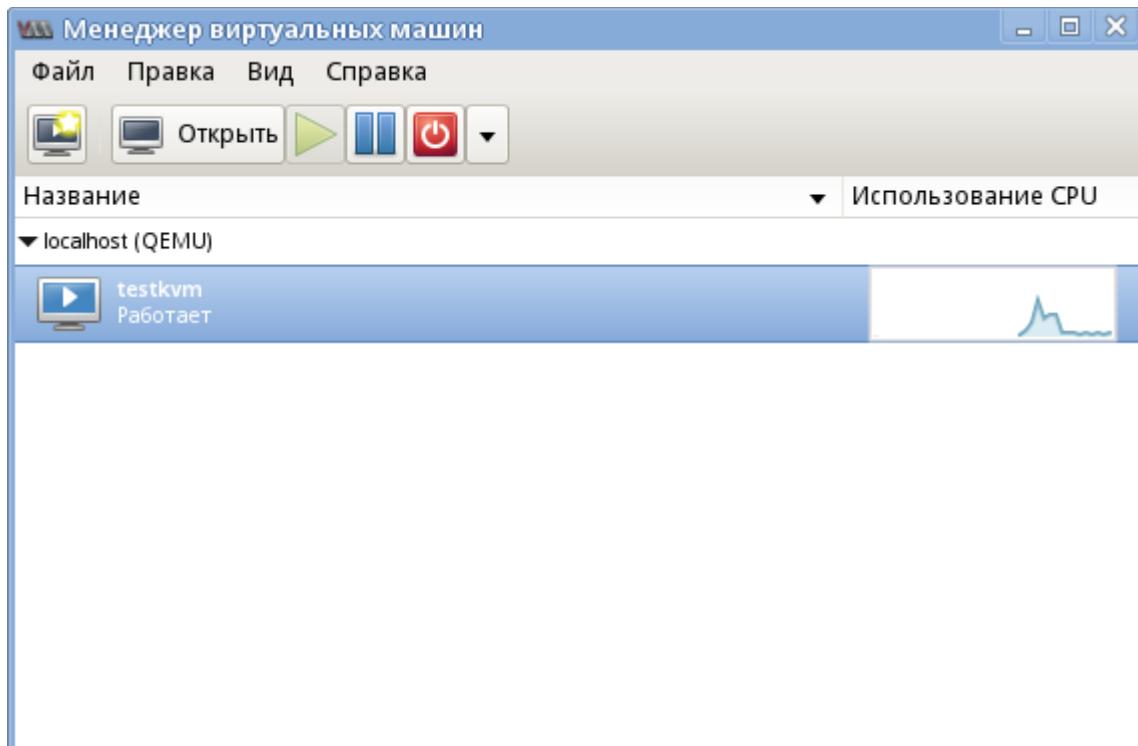
**QEMU** --- свободная программа с открытым исходным кодом для эмуляции аппаратного обеспечения различных платформ, которая может работать и без использования **KVM**, но использование аппаратной виртуализации значительно ускоряет работу гостевых систем, поэтому использование KVM в QEMU (-enable-kvm) является предпочтительным вариантом.

Первоначально разработка велась в рамках проекта **Linux KVM** (Kernel-based Virtual Machine), в котором, помимо собственно KVM (поддержки технологий аппаратной виртуализации x86-совместимых процессоров на уровне ядра Linux), разрабатывались патчи для QEMU, позволяющие QEMU использовать функциональность KVM. Однако недавно разработчики QEMU в сотрудничестве с разработчиками KVM приняли решение интегрировать поддержку KVM в основную ветку QEMU (mainline).

Используя KVM, вы можете запустить несколько виртуальных машин с немодифицированными образами Linux и Windows. Каждая виртуальная машина имеет собственную приватную среду, в которой она работает: сетевую карту, диск, графический адаптер и т.д.

Поддержка KVM вошла в ядро Linux начиная с версии 2.6.20.

**libvirt** --- это интерфейс и демон для управления виртуальными машинами различных технологий(qemu/kvm, xen, virtualbox), он позволяет удобно настраивать и манипулировать виртуальными машинами. Также к нему есть много разных сторонних приложений для управления, web-интерфейсов и т.д. Например, весьма полезным может быть графический интерфейс управления виртуальными машинами **virt-manager**.



## Установка пакетов

Внесем изменения в make.conf для поддержки архитектур гостевых машин. Если вам необходимы другие нестандартные архитектуры, то добавьте их в этот список:

```
echo 'QEMU_SOFTMMU_TARGETS="i386 x86_64"
QEMU_USER_TARGETS="i386 x86_64"' >> /etc/portage/make.conf/custom
```

Для настройки **KVM** в **Calculate Linux** нам понадобятся несколько пакетов со своими зависимостями. Аналогично данные рекомендации могут быть применимы и для **Gentoo**.

Если вы планируете использовать KVM не только для тестирования, то следует для app-emulation/qemu еще добавить USE-флаги sasl и tls для настройки безопасного подключения.

```
echo ">=net-dns/dnsmasq-2.72 script
app-emulation/qemu png virtfs xattr vnc
app-emulation/libvirt qemu udev virt-network" >>
/etc/portage/package.use/custom
emerge -a app-emulation/libvirt app-emulation/qemu
```

## Настроим параметры qemu

/etc/libvirt/qemu.conf

Позволит vnc слушать на всех адресах

```
vnc_listen = "0.0.0.0"
```

Отключим tls(если виртуальные машины используются не для тестирования, то лучше tls включить и настроить)

```
vnc_tls = 0
```

Пароль по умолчанию для vnc --- он будет использоваться в том случае, если для виртуальной машины не указан свой пароль

```
vnc_password = "XYZ12345"
```

Пользователь от которого будет запускаться qemu

```
user = "root"
```

Группа от которой будет запускаться qemu

```
group = "root"
```

Формат сохранения и дампов, gzip или другое сжатие позволит уменьшить место, занимаемое образами, но притом увеличит время сохранения образов.

```
save_image_format = "gzip"
dump_image_format = "gzip"
```

Запустим демон libvirt и добавим его в автозапуск

```
/etc/init.d/libvirtd start
rc-update add libvirtd default
```

Также для виртуальной сети нам необходим загруженный модуль tun и vhost\_net

```
modprobe -a tun vhost_net
echo "tun"
```

```
vhost_net" >> /etc/modules-load.d/kvm.conf
```

## Создание виртуальной машины

Для начала нам нужно создать образ для жесткого диска, по умолчанию он создается динамически расширяемым

```
qemu-img create -f qcow2 /var/calculate/vbox/hdd.qcow2 20G
```

Если вы хотите выделить сразу же все место, то вам нужно использовать следующую команду

```
qemu-img create -f qcow2 -o preallocation=metadata hdd.qcow2 20G
```

Если вы выделяете сразу же все место, то работать виртуальная машина будет быстрее, т.к. ей не надо будет выделять его в процессе работы.

Пример конфигурации виртуальной машины testkvm.xml. В данном примере используется следующая конфигурация 1Гб оперативной памяти, 1 ядро процессора, система 64 бита, установочный образ **Calculate Scratch Server** 14.12.1, vnc-сервер на порту 5910:

```
<domain type='kvm' id='1'>
 <features><acpi/></features>
 <name>testkvm</name>
 <description>Description of server</description>
 <memory unit='KiB'>1048576</memory>
 <vcpu placement='static'>1</vcpu>
 <os>
 <type arch='x86_64' machine='pc-1.3'>hvm</type>
 <boot dev='cdrom' />
 <boot dev='hd' />
 <bootmenu enable='yes' />
 </os>
 <devices>
 <emulator>/usr/bin/qemu-system-x86_64</emulator>
 <disk type='file' device='disk'>
 <driver name='qemu' type='qcow2' cache='writeback' io='threads' />
 <source file='/var/calculate/vbox/hdd.qcow2' />
 <target dev='vda' bus='virtio' />
 </disk>
 <disk type='file' device='cdrom'>
 <driver name='qemu' type='raw' />
 <source file='/var/calculate/linux/css-14.12.1-x86_64.iso' />
 <target dev='vdd' bus='virtio' />
 <readonly />
 </disk>
 <interface type='network'>
 <source network='default' />
 </interface>
 <graphics type='vnc' port='5910' autoport='no'>
 <listen type='address' />
 </graphics>
 </devices>
</domain>
```

Выделяемая для машины оперативная память задается строкой:

```
<memory unit='KiB'>1026608</memory>
```

Количество выделяемых CPU:

```
<vcpu placement='static'>1</vcpu>
```

Загрузочные устройства указываются в секции .

Путь к созданному образу жесткого диска задается строкой:

```
<source file='/var/calculate/vbox/hdd.qcow2' />
```

Путь к установочному образу задается в секции, его следует поменять на ваш:

```
<disk type='file' device='cdrom'>
```

Подробнее об xml-конфиге --- <http://libvirt.org/formatdomain.html>

Добавление машины на базе конфига:

```
virsh define testkvm.xml
```

Запуск машины:

```
virsh start testkvm
```

Останов машины:

```
virsh destroy testkvm
```

это экстренный останов сравнимый с отрубанием питания.

Для поддержки нормальных остановки и перезапуска в гостевой машине должна быть установлена поддержка acpi.

В случае использования Calculate Linux в качестве гостевой машины, делается это следующим образом:

```
emerge -a sys-power/acpid
```

```
rc-update add acpid boot
```

Автозапуск гостевой машины при запуске демона libvirt осуществляется следующим образом:

```
virsh autostart testkvm
```

Отключение автозагрузки:

```
virsh autostart --disable testkvm
```

Также возможен, запуск вручную без использования libvirt-демона:

```
qemu-system-x86_64 -name calculate -cpu host -smp 4 -enable-kvm -localtime -m 2048 -no-fd-bootchk -net nic,model=virtio,vlan=0 -net user,vlan=0 -drive file=hdd.img,index=0,media=disk,if=virtio -drive file=/mnt/iso/cldx-20150224-i686.iso,media=cdrom -monitor telnet:0.0.0.0:4008,server,nowait -spice port=5901,disable-ticketing -vga qxl -usb
```

## Virtio --- что это и с чем это едят

При использовании обычных виртуальных дисков и сетевых карт система работает следующим образом. В гостевую систему предоставляются копии реальных устройств. Таким образом в гостевой системе драйвера устройств преобразуют высокоуровневые запросы в низкоуровневые, виртуальная система их перехватывает, преобразует в высокоуровневые и передает уже драйверам хост-системы. При использовании virtio-устройств цепочка сокращается, virtio-драйвера ничего не преобразуют, а передают напрямую хост-системе высокоуровневые запросы, что приводит к ускорению работы виртуальных машин.

## Команды изменение привода

Для изменения образа в виртуальном приводе можно воспользоваться следующими командами.

Извлечение образа:

```
change-media guest01 vdd --eject
```

Вставка нового образа:

```
change-media guest01 vdd /pool/disc.iso
```

Общий формат команды:

```
change-media <domain> <path> [<source>] [--eject] [--insert] [--update]
[--current] [--live] [--config] [--force]
```

## Настройка сети

При базовых настройках используется виртуальная сеть недоступная извне.

Доступ по ip может быть осуществлен с компьютера, на котором поднят KVM. Изнутри доступ происходит через NAT.

Вариантов настройки сети есть множество и это отдельная тема, которую можно разбирать очень и очень долго и ориентироваться надо на конкретный случай применения. Возможные варианты настройки сети:

- NAT Based --- это вариант по умолчанию. Внутренняя сеть, предоставляющая доступ к внешней сети с автоматическим применением NAT. В приведенном выше примере конфигурации используется именно этот вариант.
- Routed --- Аналогично предыдущему внутренняя сеть, предоставляющая доступ к внешней сети, но без NAT. Предполагает дополнительные настройки таблиц маршрутизации во внешней сети.
- Изолированная IPv4/IPv6 сеть.
- Bridge --- Подключение типа мост (Позволяет реализовать множество различных конфигураций, в том числе и с назначение IP из реальной сети).
- Перенаправление одной PCI сетевых карт хост-машины на гостевую машину.

Если вы хотите настраивать сеть через bridge, то вам следует пересобрать ядро, включив следующие параметры:

```
CONFIG_MACVLAN=m
CONFIG_MACVTAP=m
CONFIG_BRIDGE_NF_EBTABLES=m
CONFIG_BRIDGE_EBT_MARK_T=m
CONFIG_BRIDGE_EBT_T_NAT=m
```

Подробно с возможностями настройки сети можно ознакомиться на страницах:

- <https://libvirt.org/formatnetwork.html>
- <http://wiki.libvirt.org/page/Networking>

## Gitolite + git настройка

**Gitolite** --- удобное средство, позволяющее управлять [Git](#) репозиториями, используя один пользовательский аккаунт, управляем доступом пользователей к репозиториям на основе ssh-ключей, в отличии от [gitosis](#) позволяет более гибко настраивать доступ к репозиториям.

Gitolite управляет несколькими репозиториями под одной учетной записью пользователя, с использованием SSH ключей для идентификации пользователей. Конечным пользователям не нужны учетные записи на сервере, они будут входить через один общий аккаунт, который не позволит им запускать произвольные команды.

## Установка gitolite

Если пакет замаскирован, размаскируйте его.

```
emerge dev-vcs/gitolite
```

После установки будет создан пользователь git (пароль отсутствует, домашняя директория /var/lib/gitolite/)

## Настройка gitolite

### Создаем ключ для root

```
su
ssh-keygen -t rsa
```

Будут созданы два файла:

```
/root/.ssh/id_rsa.pub
/root/.ssh/id_rsa
```

Публичный и закрытый rsa ключи

### Создаем репозиторий с настройками

Копируем открытый ключ

```
cp /root/.ssh/id_rsa.pub /tmp/id_rsa.pub
```

Инициируем gitolite

```
su git
cd
gitolite setup -pk /tmp/id_rsa.pub
```

Удаляем из /tmp открытый ключ

```
rm /tmp/id_rsa.pub
```

Если планируется использовать git-web, для того чтобы, репозиторий с настройками был не виден в web изменим права на директорию

```
chmod 700 /var/lib/gitolite/repositories/gitolite-admin.git
```

## Настраиваем репозиторий для пользователя

Переходим в /tmp

```
cd /tmp
```

Клонируем директорию с настройками

```
git clone git@имя_сервера:gitolite-admin.git
```

Переходим в директорию с настройками

```
cd gitolite-admin
```

Копируем пользовательский открытый ключ в директорию /tmp/gitolite-admin/keydir

```
scp root@имя_клиентского_компьютера:/home/имя_пользователя/.ssh/id_rsa.pub
.keydir/имя_пользователя@имя_клиентского_компьютера.pub
```

Настраиваем новый репозиторий для пользователя имя\_пользователя

```
vi conf/gitolite.conf
```

```
repo gitolite-admin
 RW+ = id_rsa

repo project1
 RW+ = testuser@remotehost
```

Права могут быть указаны следующим образом:

- R - только чтение
- RW - чтение добавление коммитов, запрещен rewind (push --force)
- RW+ - полный доступ
- RWC - возможность создавать ветку
- RWD - возможность удалять ветку
- - запретить запись объекты доступа могут быть:
  - master@ - ветки
  - ref/tags - теги (версии)
  - NAME/имя\_файла - имена файлов/каталогов

Для создания группы используется следующий синтаксис

```
@groupname = user1 user2 user3
```

Также этот синтаксис используется для группировки веток

```
@importantbranches = master$ developer$ test$
```

```
repo testing
 RW @importantbranches = @groupname
 - @importantbranches = @groupname
 RW+ = @groupname
```

Делаем commit для настроек

```
git commit -am "Granted full access for testuser@remotehost to project1"
```

Записываем репозиторий с настройками

```
git push origin master
```

## Создаем новый репозиторий

В отличии от gitosis в gitolite можно предварительно не создавать репозиторий через --bare init, достаточно добавить его в конфигурационный файл.

Этот репозиторий будет доступен пользователю на его компьютере через ssh

```
git clone ssh://git@имя_сервера/project1.git
```

без ввода пароля (по открытому ключу)

## Проверка sshd

Демон sshd должен быть запущен.

Должен быть разрешен доступ по ssh пользователю git файл /etc/ssh/sshd\_config  
AllowUsers git

## Настройка git-daemon

vi /etc/conf.d/git-daemon

```
GITDAEMON_OPTS="--syslog --port=9418 --base-
path=/var/lib/gitolite/repositories/ --export-all"
GIT_USER="apache"
```

Если будут проблемы при перезапуске меняем номер порта --port=9418 на другой номер перезапускаем демона git, меняем обратно перезапускаем демона git.

/etc/init.d/git-daemon restart

## Проблема перезапуска git-daemon

При перезапуске демона может возникнуть проблема повторного открытия порта, в /var/log/message она записывает как

```
git-daemon[pid]: unable to allocate any listen sockets on host (null) port
9418
```

Чтобы ее избежать необходимо добавить в GITDAEMON\_OPTS параметр --reuseaddr

## Настройка прав на файлы репозиториев

При сохранение файлов репозиториев gitolite присваивает их пользователю и группе git. Права же устанавливаются согласно параметру UMASK в файл /var/lib/gitolite/.gitolite.rc. Это может быть важно, если git-daemon запускается не под пользователем git.

## Создание пользовательского ключа

В сеансе пользователя, на пользовательском компьютере.

ssh-keygen -t rsa

В пользовательской директории .ssh будут созданы два ключа

```
id_rsa
id_rsa.pub
```

закрытый и публичный (pub)

Если публичный ключ записан в директорию keys репозитория настроек gitolite на сервере, а также в файле conf/gitolite.conf настроены права к репозиториям то пользователь получит доступ к репозиториям на сервере.

## Примеры прав доступа к репозиториям

Разрешить группе developers производить любые действия с репозиторием кроме модификации ветки master

```
repo myrep
- master$ = @developers
RW+ = @developers
```

Запретить `rewind` для ветки `master`, (в целом это правило означает что группа `@developer` может добавлять коммиты и только в ветку `master`).

```
repo myrep
 RW master$ = @developers
```

Разрешить группе `developers` выполнять любые действия с репозиторием, кроме создание тегов, теги может создавать только `maindeveloper`.

```
repo myrep
 RW+ refs/tags = maindeveloper
 - refs/tags = @developers
 RW = developers
```

## Резервное копирование

Для надежности, обязательно используйте регулярное резервное копирование данных, даже если вы используете [raid-массив](#).

### Резервное копирование настроек

Сохранение резервной копии учётных записей пользователей, а также настроек серверов, выполненных при помощи утилит [Calculate 2](#), выполняется командой:

```
c1-backup
```

Вы можете настроить ежедневное сохранение резервной копии, создав в директории `/etc/cron.daily/` файл со следующим содержимым:

```
#!/bin/bash
c1-backup
```

Не забудьте присвоить файлу права на выполнение.

### Резервное копирование данных

Рассмотрим случай, когда для хранения данных на сервере используются три жестких диска, подмонтированных следующим образом:

```
#cat /etc/fstab
...
/dev/sda5 /var/calculate xfs noatime 0 0
/dev/sdb1 /var/calculate/server-data xfs noatime 0 0
/dev/sdc1 /var/calculate/server-data/samba/share xfs noatime 0 0
...
```

В примере мы используем резервное копирование файлов на другой сервер, выполняющий функции хранилища копии данных. Backup-сервер также располагает жесткими дисками не меньшей ёмкости. Соответственно копировать информацию нужно будет с каждого диска по отдельности.

Большую часть времени Backup-сервер выключен и включается только во время копирования данных. Для настройки функции пробуждения сервера, воспользуйтесь следующим [описанием](#).

Пример скрипта синхронизации данных пользователей с Backup-сервером.

```
#!/bin/bash

#backup-сервер
BACKUP_IP=192.168.0.250
```

```
BACKUP_MAC=01:14:7b:10:3a:6a
```

```
#разбудим Backup-сервер
wol $BACKUP_MAC
```

```
#ждем 5 минут пока проснется
sleep 300
```

```
#синхронизируем данные с трех разделов
rsync -e ssh --progress -aAzuhvSx --compress-level=9 --delete-after
/var/calculate/ root@$BACKUP_IP:/mnt/backup1
rsync -e ssh --progress -aAzuhvSx --compress-level=9 --delete-after
/var/calculate/server-data/ root@$BACKUP_IP:/mnt/backup2
rsync -e ssh --progress -aAzuhvSx --compress-level=9 --delete-after
/var/calculate/server-data/samba/share/ root@$BACKUP_IP:/mnt/backup3
```

```
#выключим до следующего пробуждения
ssh -o 'StrictHostKeyChecking=no' root@$BACKUP_IP halt
```

В приведенном примере копируются три раздела */var/calculate*, */var/calculate/server-data* и */var/calculate/server-data/samba/share* с параметром "x", что позволяет при копировании не выходить за пределы монтированного раздела. Приведем пример наиболее популярных ключей команды *rsync*.

- а - включает -rlptgoD
- А - переносить права ACL
- е --- заменяет протокол с rsh на ssh
- progress --- вывод хода процесса работы на терминал
- l --- пересоздание symlinks, это значит, что символические ссылки будут так же переноситься
- z --- использовать сжатие
- u --- update, обновление, он будет пропускать файлы которые новей, чем на удалённом сервере
- о --- установить владельца конечного файла таким же, как и у исходного
- g --- установить группу конечного файла таким же, как и у исходного
- t --- передача времени модификации и его обновление на удаленной системе. Этот ключ должен быть установлен для точной синхронизации
- h --- вывод информации на терминал в удобном для чтения (human-readable) виде
- v --- verbose, вывод сообщений в терминал
- р - сохранять права доступа
- г --- рекурсивный режим
- S - корректно обрабатывать разреженные (sparse) файлы.
- х - не выходить за пределы ФС (например при копировании /)
- н --- отладочный режим
- compress-level --- уровень сжатия
- delete-after --- удалять файлы, которые не были найдены на удалённом сервере, "-after" означает, что удалить их нужно, только после окончания синхронизации. Так-же есть *delete-before*, *delete-during*, *delete-excluded* и просто *delete*

## Открытый ключ

В приведенном примере файлы передаются посредством *ssh*, соответственно перед каждой передачей будет запрашиваться пароль *root*. Вы можете создать открытый ключ и скопировать его на Backup-сервер, в этом случае пароль запрашиваться не будет.

Для этого создайте на сервере *rsa* ключ, выполнив:  
*ssh-keygen*

```
Скопируйте ключ пользователю root Backup-сервера
ssh-copy-id -i /root/.ssh/id_rsa.pub BACKUP_IP
```

## Отладочный режим

Перед тем как запустить скрипт на рабочей машине, опробуйте всё в отладочном режиме, для этого используется ключ "-n". В этом случае, rsync не будет менять или удалять файлы, но покажет весь ход работы.

## Исключение файлов

Rsync может как исключать, так и включать файлы по заданному шаблону. Например "---exclude \*.run" исключит все файлы с расширением *run*.

# 6. Настройка рабочей станции

1. [Переход на Linux](#)
2. [Подключение к серверу каталогов](#)
3. [Хранение пользовательских настроек](#)
4. [Установка шаблонов пользовательского окружения](#)
5. [Реализация кэширования NSS для доменной машины](#)

## Переход на Linux

6. [Переход на Linux](#)
7. [Настройка операционных систем](#)
8. [Calculate Directory Server](#)
9. [Calculate Linux Desktop](#)
10. [Windows Workstation](#)
11. [Работа с терминальным Windows-сервером](#)

## Настройка операционных систем

При помощи двух пакетов - *calculate-client* и *calculate-server* - вы можете построить гетерогенную сеть с Windows- и Linux-клиентами. Для того, чтобы использовать все предлагаемые возможности, используйте последние версии [Calculate Directory Server](#) с предустановленным пакетом *calculate-server* и [Calculate Linux Desktop](#), включающим *calculate-client* со всеми необходимыми для работы пакетами.

## Calculate Directory Server

Установка и настройка контроллера домена.

1. Загрузите LiveCD [Calculate Directory Server](#) и запишите образ на CD.
2. Загрузитесь с LiveCD. Теперь вы можете проверить работу системы и [установить ее](#) на жесткий диск.
3. Выполните [настройку сети](#).
4. [Настройте Samba-сервер](#) и другие необходимые сервисы.
5. Добавьте пользователей домена.
6. Установите пароль служебным пользователям *client* и *admin*, которые понадобятся для ввода соответственно Linux- и Windows- клиентов в домен, выполнив:

```
cl-passwd --smb client samba
cl-passwd --smb admin samba
```

1. Настройте использование *distfiles* клиентами домена

```
cl-groupadd --gid 250 -f portage unix
cl-usermod -a portage client unix
```

```
chmod 0775 /var/calculate/remote
chmod -R 2775 /var/calculate/remote/distfiles
chown -R root:portage /var/calculate/remote/distfiles
```

## Calculate Linux Desktop

Установка Linux-клиентов.

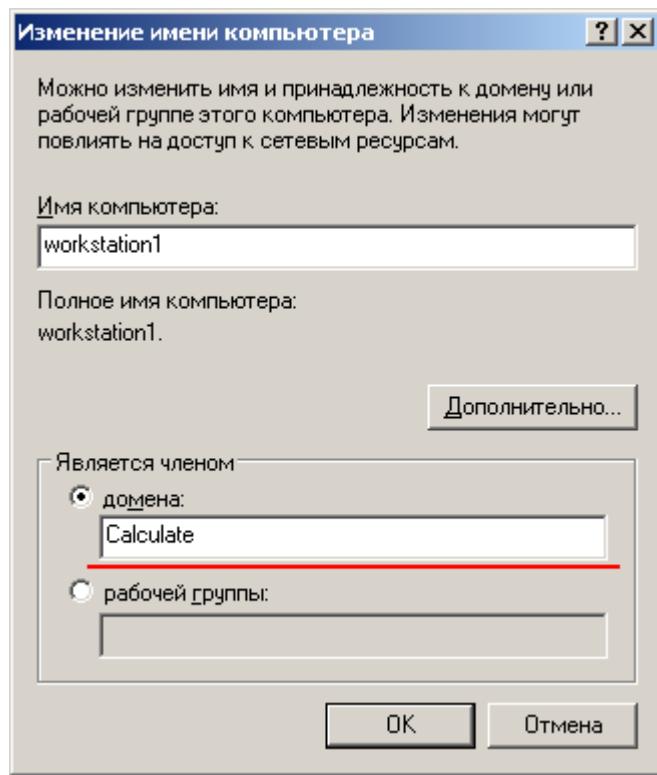
1. [Загрузите](#) LiveDVD [Calculate Linux Desktop](#) и запишите образ на DVD.
2. Загрузитесь с LiveDVD. Теперь вы можете проверить работу системы и [установить её](#) на жесткий диск.
3. [Ведите компьютер в домен Calculate Directory Server](#). При входе в домен будут доступны:
4. сетевой диск *Home*, используемый для хранения личных файлов пользователя;
5. сетевой диск *Disks*, используемый для хранения общих файлов;
6. сетевой диск *FTP*, используемый для быстрого доступа к FTP (если FTP-сервер сконфигурирован).



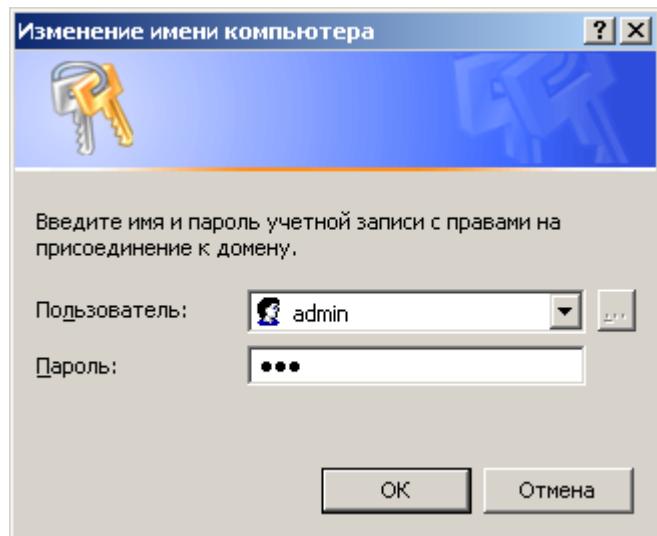
## Windows Workstation

Для присоединения Windows-клиентов в домен [Calculate Directory Server](#) выполните следующие действия:

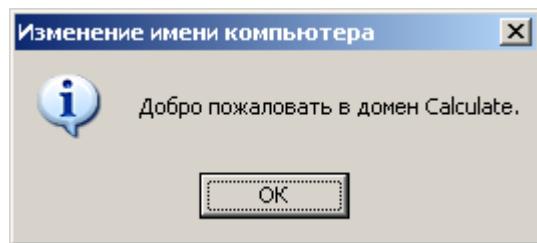
1. Откройте *Система* (*Пуск* » *Настройка* » *Панель управления* » *Система*), перейдите на вкладку *Имя компьютера* и нажмите кнопку *Изменить...*.



1. В открывшемся окне отметьте **Является членом домена**, впишите имя NetBIOS-группы, в которую входит сервер домена (по умолчанию - *Calculate*), и нажмите **OK**.
2. После выполненных действий откроется окно для ввода имени и пароля. Впишите в поле логин *admin* и укажите пароль, который был задан при настройке *Calculate Directory Server* пользователю *admin*.



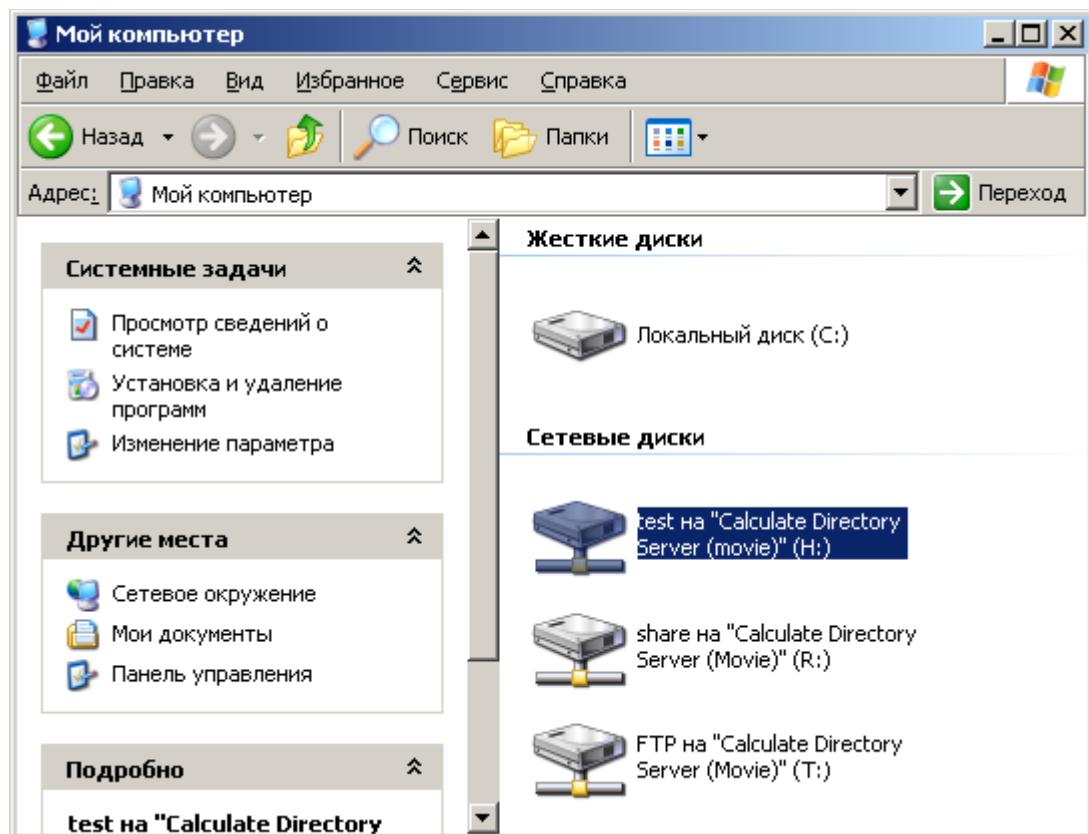
1. При удачном подключении вы получите сообщение о вводе компьютера в домен. Поздравляем! Теперь компьютер следует (в лучших традициях Windows) перезагрузить.



При первом подключении к домену может появиться ошибка, в этом случае повторите вход, введя пароль пользователя *admin*.

1. После перезагрузки в окне входа в систему не забудьте указать ваш домен. При входе в домен будут доступны:
2. сетевой диск *H:*, используемый для хранения личных файлов пользователя;

3. сетевой диск *R:*, используемый для хранения общих файлов;
4. сетевой диск *T:*, используемый для быстрого доступа к FTP (если FTP-сервер сконфигурирован).



## Работа с терминальным Windows-сервером

Настройку Windows-сервера мы опустим. После настройки введите сервер в домен [Calculate Directory Server](#) - аналогично тому, как это делается для *Windows Workstation*.

Особого внимания заслуживает процесс запуска Windows-приложений в среде *CLD*.

Для примера, создадим иконку запуска *Photoshop*. Важно, чтобы окно занимало весь экран и в то же время не заезжало под верхнюю панель.

Управлять иконками можно на сервере: для этого создайте файл с иконкой в шаблоне пакета [calculate-client](#) в подмонтированном ресурсе `/var/calculate`.

```
/var/calculate/remote/client-
profile/always/.local/share/applications/photoshop.desktop
```

Файл должен быть следующего содержания:

```
[Desktop Entry]
Version=1.0
Name=Adobe Photoshop
GenericName=графический редактор
Comment=
Exec=/usr/bin/keyexec rdesktop "-s 'C:\\Program Files\\Adobe\\Adobe Photoshop CS3\\photoshop.exe' -d calculate -a 16 -g #-hr_x11_width-#x#-sum(h,hr_x11_height-31)-# -T 'Adobe Photoshop' -S standard -zNDKE -p -winserver.localnet.org"
Icon=photoshop
Terminal=false
StartupNotify=true
Type=Application
Categories=Graphics;
```

Примечание:

- Имя сервера *winserver.localnet.org* нужно заменить на имя вашего сервера.
- Иконка *photoshop* входит в состав *Calculate Linux Desktop*.
- Размер окна вычисляется в [функции sum](#), где 32 - высота верхней панели.
- Утилита [keyexec](#) позволяет запускать приложения по хранимому в ключах ядра паролю.
- Если вы предпочитаете хранить пароль доступа к *1C* в иконке запуска, воспользуйтесь [функцией load](#) для считывания и подстановки содержимого файла в шаблон.

## Подключение к серверу каталогов

Наибольший интерес пакет *calculate-client* представляет при наличии сервера директорий. В этом случае вы можете хранить все настройки централизованно. Для взаимодействия с сервером на последнем должен быть установлен пакет *calculate-server*. Вы можете воспользоваться сервером [Calculate Directory Server](#), в состав которого входит пакет *calculate-server*.

Сеанс пользователя в этом случае настраивается с учетом работающих на сервере приложений. Для доступа к настройкам рабочую станцию нужно ввести в домен.

*Мы применяем терминологию, похожую на Windows NT, т.к. во многом принцип действия схож. Вы можете подключать к серверу как Windows машины, так и Linux. Обе системы получают доступ к общим ресурсам сервера, сетевым службам, а также хранят на нем свои настройки - окружения рабочих столов пользователей.*

**Для ввода в домен потребуется выполнить следующие действия:**

1. Выполните настройки *samba* на сервере по [описанию](#).
2. Установите пароль пользователя *client* на сервере, выполнив:

```
cl-passwd --smb client samba
```

1. Введите компьютер в домен. Для этого регистрируемся на клиентской машине как *root* и выполняем:

```
cl-client HOST
```

Потребуется ввод пароля пользователя *client*. Вместо сетевого имени сервера (в примере *HOST*, проверьте доступность, выполнив *ping HOST*) можно указать его IP адрес.

Введенный пароль кэшируется на клиентской системе в файле */var/calculate/calculate.env*:

```
[client]
cl_remote_host = HOST
cl_remote_pw = password
```

**При успешном входе в домен программа выполнит следующие действия:**

- Подмонтирует сетевой ресурс */var/calculate/remote*.
- Подключит пользователей сервера директорий, настроив файлы */etc/pam.d/system-auth*, */etc/nsswitch.conf* и др.
- Подготовит */home*, монтируя поверх локальную директорию */var/calculate/client-home*, для исключения конфликтов с локальными пользователями.
- Пропишет себя в автозагрузку, чтобы при загрузке выполнять проверку доступности сервера и настройку системы.

## Хранение пользовательских настроек

Одной из наиболее значимых возможностей пакета *calculate-client* является централизованное хранение настроек пользователей.

При входе в систему и при завершении сеанса выполняется программа *cl-sync*, которая синхронизирует настройки сеанса пользователя с сервером. Это происходит только в том случае, если вы зарегистрированы в домене. Передача файлов осуществляется при помощи программы *rsync*, которая оптимизирует трафик, передавая только изменённые файлы.

Помимо синхронизации, *cl-sync* монтирует домашнюю директорию пользователя в */home/\$USER/Home*, а также сетевые ресурсы в */home/\$USER/Disk*.

## Установка пользовательского окружения

### Введение

При установке системы на предприятии, либо при большом количестве пользователей, может встать вопрос об унификации интерфейса пользователя, добавлении необходимых настроек. Редактировать настройки каждого сеанса - достаточно трудоемкое и неблагодарное занятие. Здесь мы расскажем о возможностях пакета *calculate-client* по настройке рабочего стола пользователя на примере KDE4.

### Шаблоны настроек

Утилиты [Calculate](#) в своей работе используют шаблоны. Это достаточно универсальный и мощный механизм для настройки программного обеспечения. При помощи шаблонов можно настроить систему, установить и настроить сервер или изменить внешний вид рабочего стола.

Ознакомьтесь с [подробным руководством](#) по возможностям шаблонов. В этой статье мы покажем несколько примеров по возможностям настройки приложений пользователя.

### Размещение

При наличии [Calculate Directory Server](#), рабочий стол пользователя может быть настроен с учётом особенностей работы предприятия. При [вводе Linux ПК в домен](#), при помощи утилиты *cl-client*, монтируется сетевой диск в директорию */var/calculate/remote*. Это отличное место для размещения шаблонов настройки профилей пользователей.

Для начала посмотрите примеры шаблонов из состава пакета *calculate-client* в директории */var/lib/layman/calculate/profiles/templates/client*. Директории *domain* и *undomain* содержат шаблоны ввода рабочей станции в домен и вывода.

По умолчанию в утилитах *calculate-client* 2.1.x может быть неограниченное количество директорий в шаблоне, разбивающих настройки на логические составляющие, обрабатываемые в алфавитном порядке. Внутри любой директории может быть служебный файл *.calculate\_directory*, определяющий условия применения шаблонов этой директории. Если этого файла нет, шаблон будет применяться при каждом входе в систему. В случае если там будет ложное условие проверки, шаблон из директории применяться не будет.

Пример создания шаблона *MyFirm* для клиентов [Calculate Linux Desktop](#):

```
mkdir -p /var/calculate/remote/templates/client/MyFirm
```

Если вы используете Calculate Linux Desktop XFCE, здесь и далее в примерах замените директорию шаблона с CLD на CLDX.

Создайте файл *.calculate\_directory* в директории шаблона, если вы хотите добавить условия применения шаблона *MyFirm*. Например, добавить условие применения этого шаблона только при первом сеансе можно, добавив следующее содержимое в этот файл:

```
Calculate cl_pass_step==first
```

Если вы используете Calculate Linux Desktop без домена, вы можете создать описанные директории локально либо на сетевом диске.

# Использование

Утилиты из пакета [calculate-client](#) имеют богатые возможности по настройке рабочего стола [KDE4](#). В зависимости от приложения, создается один или более файлов шаблона. Большинство настроек хранятся в директории `.kde4/share/config`. Файлы в этой директории имеют название схожее с именем приложения и заканчиваются на `rc`. Дополнительные настройки могут храниться в директории приложения, расположенной в `.kde4/share/apps`. Директория называется также по имени приложения, но без окончания `rc`.

Для создания шаблона, просто создайте в директории удаленного шаблона файл с тем же путем.

Обратите внимание, что интерпретатор должен правильно определить формат файла, поэтому в первой строке шаблона вы должны указать его формат. Для большинства конфигурационных файлов KDE4 это будет `kde`:

```
Calculate format=kde
```

Прочитав формат `kde`, интерпретатор выполнит объединение файла шаблона с существующим файлом настроек. Таким образом, в шаблоне вам достаточно указать только необходимые изменения.

Обратите внимание: отсутствие заголовка в файле шаблона, в зависимости от содержимого, будет восприниматься как текстовый либо бинарный файл. Такой файл будет попросту переписывать текущий.

Следует также не забывать о том, что в формате `kde` все переменные расположены в областях действия имен. Области имеют названия, записанные в квадратных скобках. В шаблонах обязательно следует указывать, в какой области находятся заменяемые переменные.

Попробуем разобраться на примерах.

## Примеры

### Стартовая страница браузера

В шаблоне `MyFirm` отредактируем настройки браузера `Konqueror`, изменив адрес домашней страницы пользователя.

Создадим путь к файлу в шаблоне:

```
mkdir -p /var/calculate/remote/templates/client/MyFirm/.kde4/share/config/
```

в котором создадим файл `konquerorrc` со следующим содержимым:

```
Calculate format=kde
[UserSettings]
#Домашняя страница
HomeURL=http://www.google.com
```

Первая строка служебная, в ней мы описываем формат конфигурационного файла. По умолчанию созданный нами шаблон объединяется с существующим, меняя значения переменных. Таким образом, после захода в сеанс пользователь всегда в качестве домашней страницы будет попадать на страницу поисковика Google, как бы он настройки ни менял.

Но вот страница, которая отображается в браузере первой, будет оставаться той же, так как она настраивается в другом месте. Для ее изменения создадим следующий путь:

```
mkdir -p
/var/calculate/remote/templates/client/MyFirm/.kde4/share/apps/konqueror/p
rofiles
```

в котором создадим файл `webbrowsing` со следующим содержимым:

```
Calculate format=kde
[Profile]
```

```
#просмотр домашней страницы при старте Konqueror
View0_URL=http://www.google.com
```

Теперь при запуске браузера пользователь будет видеть страницу Google. При нажатии на кнопочку домашней страницы в браузере также будет отображаться страница Google.

Обратите внимание, что в обоих примерах указываются области действия переменных [UserSettings] и [Profile]. Важно правильно указать их, разместив переменные после их объявления; этого требует синтаксис конфигурационного файла.

## Прокси сервер

Приведем еще один пример. Допустим, вам нужно изменить прокси сервер. Создайте следующий путь:  
`mkdir -p  
/var/calculate/remote/templates/client/MyFirm/single/.kde4/share/config`

в который поместите файл `kioslaverc` со следующим содержимым:

```
Calculate format=kde
[Proxy Settings]
AuthMode=0
NoProxyFor=
Proxy Config Script=
ProxyType=1
ReversedException=false
ftpProxy=http://proxy.mydomain.ru:8080
httpProxy=http://proxy.mydomain.ru:8080
httpsProxy=http://proxy.mydomain.ru:8080
```

## Реализация кэширования NSS для доменной машины

**LDAP** - относительно простой протокол, использующий TCP/IP и позволяющий производить операции аутентификации (bind), поиска (search) и сравнения (compare), а также операции добавления, изменения или удаления записей. Обычно LDAP-сервер принимает входящие соединения на порт 389 по протоколам TCP или UDP.

**NetworkManager** - программа для управления сетевыми соединениями в Linux. Для использования NetworkManager в графическом интерфейсе, существует программа nm-applet, которая соответствует стандарту freedesktop.org System Tray Protocol, включая KDE, XFCE, GNOME. NetworkManager использует D-Bus, udev и PolicyKit. Компоненты взаимодействуют через D-Bus.

**D-Bus** - система межпроцессного взаимодействия, которая позволяет приложениям в операционной системе общаться друг с другом.

**PolicyKit** - набор инструментов разработки для контроля системных привилегий в Unix-подобных операционных системах. Он даёт возможность непrivилегированным процессам общаться с привилегированными. Для авторизации сетевых пользователей Calculate Linux Desktop использует LDAP сервер. Во время работы доменного пользователя при продолжительном отключении сети могут возникнуть следующие проблемы:

- недоступен LDAP;
- нет возможности авторизовать пользователя (авторизация через LDAP);
- если сеанс пользователя заблокирован - его не разблокировать;
- нет возможности восстановить сетевое соединение (действие с сетью требуют информацию о пользователе).

Эти же сложности могут возникнуть и при выходе компьютера из спящего режима.

## Использование Nscd

Получение информации о пользователях осуществляется при помощи диспетчера службы имён **NSS** (Name Service Switch). Порядок и источники данных описываются в файле `/etc/nsswitch.conf`. В случае введенной в домен машины источника данных два: `files (/etc/passwd, /etc/group)` и `LDAP`. Для того, чтобы NSS информация из LDAP не пропадала при отключении сети, в Calculate Linux Desktop начиная с версии 13.11 работает демон `Nscd`.

**Nscd** - это служба, которая кэширует запросы службы имён. Она содержит два кэша на каждую категорию: попаданий (найденный элемент) и промахов (ненайденные элементы). Каждый кэш имеет для своих данных отдельный TTL (время жизни). Эти параметры настраиваются в `/etc/nscd.conf`. Для того, чтобы программа использовала кэширование, `nscd` должен быть запущен перед ней. Таким образом NSS кэш может решить проблему отключения сети, благодаря тому, что информация о пользователе будет получаться из кэша, пока он не устареет, даже если нет соединения с LDAP.

Ранний запуск `nscd` порождает проблему связанную с аутентификацией доменного пользователя до поднятия сети. В `pam.d/system-auth` используется модуль `pam_client` для ожидания соединения с LDAP если введен пользователь и пароль до поднятия сети. Проблема возникает из-за кэша промахов: менеджер входа в сеанс (`lightdm`) успевает до вызова `pam_client` запросить от NSS информацию об аутентифицируемом пользователе, а так как соединения с LDAP на этот момент нет, то пользователь попадает в кэш промахов и в течении времени жизни этого кэша программы будут получать результат о том, что пользователь не существует, даже если LDAP уже доступен. Для решения этой проблемы, при запуске демона кэш промахов отключён. Включается он уже в конце загрузки системы службой "local". Для реализации этого, в системе содержатся два файла настроек `/etc/nscd.conf` (без кэша), `/etc/nscd-cache.conf` (с кэшем).

Для того, чтобы можно было переводить компьютер в спящий режим на несколько часов/дней и при этом не терять закэшированную информацию после пробуждения, TTL кэша попаданий установлено на 7 дней. В этом случае возникает проблема актуальности записей в кэше: пользователь может быть удален из LDAP, но при этом он будет присутствовать в кэше. Для решения этой проблемы кэш обновляется при входе доменного пользователя в сеанс, при пробуждении компьютера из спящего/ждущего режима, а также по cron-у раз в три часа (интервал в часах может быть изменен при помощи переменной `client.cl_client_nscd_cache`), при условии, что LDAP сервер доступен. Для принудительного вызова обновления кэша можно использовать команду `nscd-refresh`.

## Назначение pam\_client

`Pam_client` предназначен для ожидания LDAP сервера перед аутентификацией и выполняет следующие действия:

1. на основании файлов `/etc/passwd` и `/var/lib/calculate/calculate-client/cache/passwd` определят является ли аутентифицируемый пользователь локальным;
2. если пользователь не локальный, модуль ждет запуск службы `client`;
3. после этого проверяет доступность LDAP сервера.

Определение локального пользователя необходимо для того, чтобы не выполнять ожидание если введен неверный пароль для локального пользователя. Ожидание службы `client` необходимо, для того, чтобы система была загружена и настроена на работу с доменом, перед тем как доменный пользователь будет авторизован и войдет в сессию. Ожидание LDAP сервера используется при разблокировании сеанса пользователя, после выхода из спящего режима, пока сеть не поднялась.

## 7. Настройка сети в Calculate

1. [Настройка сети Gentoo-way](#)
2. [Настройка сети Calculate-way](#)

# Настройка сети

Настройки сети находятся в файле `/etc/conf.d/net`.

## Начальная настройка

Перед тем, как настраивать конфигурационный файл, нужно убедиться, что в директории `/etc/init.d/` присутствует символическая ссылка на `/etc/init.d/net.lo` с именем конфигурируемого интерфейса (в примерах упоминается сетевой интерфейс `eth0`).

Если настройки сети получаем по *DHCP*, то конфигурационный файл будет иметь вид:

```
config_eth0="dhcp"
```

Если используем статический адрес, конфигурационный файл будет иметь вид:

```
#для интерфейса eth0 прописываем ip-адрес из 24-й сети
config_eth0="192.168.0.7/24"
#указываем шлюз который будет использоваться в сети по умолчанию
routes_eth0="default via 192.168.0.1"
```

Если настройки не прописаны в конфигурационном файле, то будет предпринята попытка получить настройки сети по *DHCP*.

Также при запуске системы можно обнаружить такого рода сообщение:

```
"Bringing up interface eth0
Starting ifplugd on eth0
Backgrounding.....
WARNING: net.eth0 has started, but is inactive"
```

Не пугайтесь. Сеть продолжает подниматься, просто не тормозит загрузку системы. В некоторых случаях это удобно, а в некоторых нет. Те, кому подобная ситуация не нравится, могут дописать в `/etc/conf.d/net` такую строку:

```
modules_eth0="!plug"
```

Но помните, при следующем запуске придется дожидаться окончания загрузки сети.

## Расширенные настройки сети

Иногда при наличии одной сетевой карты существует необходимость работать в нескольких сетях. Это можно реализовать, присвоив несколько адресов одному сетевому интерфейсу. Например:

```
#основные настройки получаем по DHCP, а IP присваиваем статически
config_eth0="dhcp" "10.0.0.17/24"
#говорим, что пакеты для сети 10.0.0.0/24 пересыпать не через шлюз,
#полученный по DHCP, а через 10.0.0.1
routes_eth0="10.0.0.0/24 via 10.0.0.1"
```

Для получения основных настроек сети не обязательно использовать *DHCP* - они могут быть заданы и статически:

```
#прописываем несколько статических адресов
config_eth0="192.168.0.17/24" "10.0.0.17/24"
#указываем основной шлюз и дополнительный
routes_eth0="default via 192.168.0.1" "10.0.0.0/24 via 10.0.0.1"
```

*MTU* для интерфейса прописываем таким способом (актуально для *DSL*-модемов, в основном используют значение *1000*):

```
mtu_eth0="1500"
```

Настройки *DNS* для интерфейса:

```
#указываем используемый в сети домен
dns_domain_eth0="your.domain"
указываем список DNS серверов в порядке их обхода
dns_servers_eth0="192.168.0.2 192.168.0.3"
```

Если данные настройки отсутствуют, то настройки DNS берутся из */etc/resolv.conf*.

Настройка синхронизации с сервером времени (*NTP*):

```
#перечисляем список серверов для синхронизации
ntp_servers_eth0="192.168.0.2 192.168.0.3"
```

Изменение *MAC-адреса*:

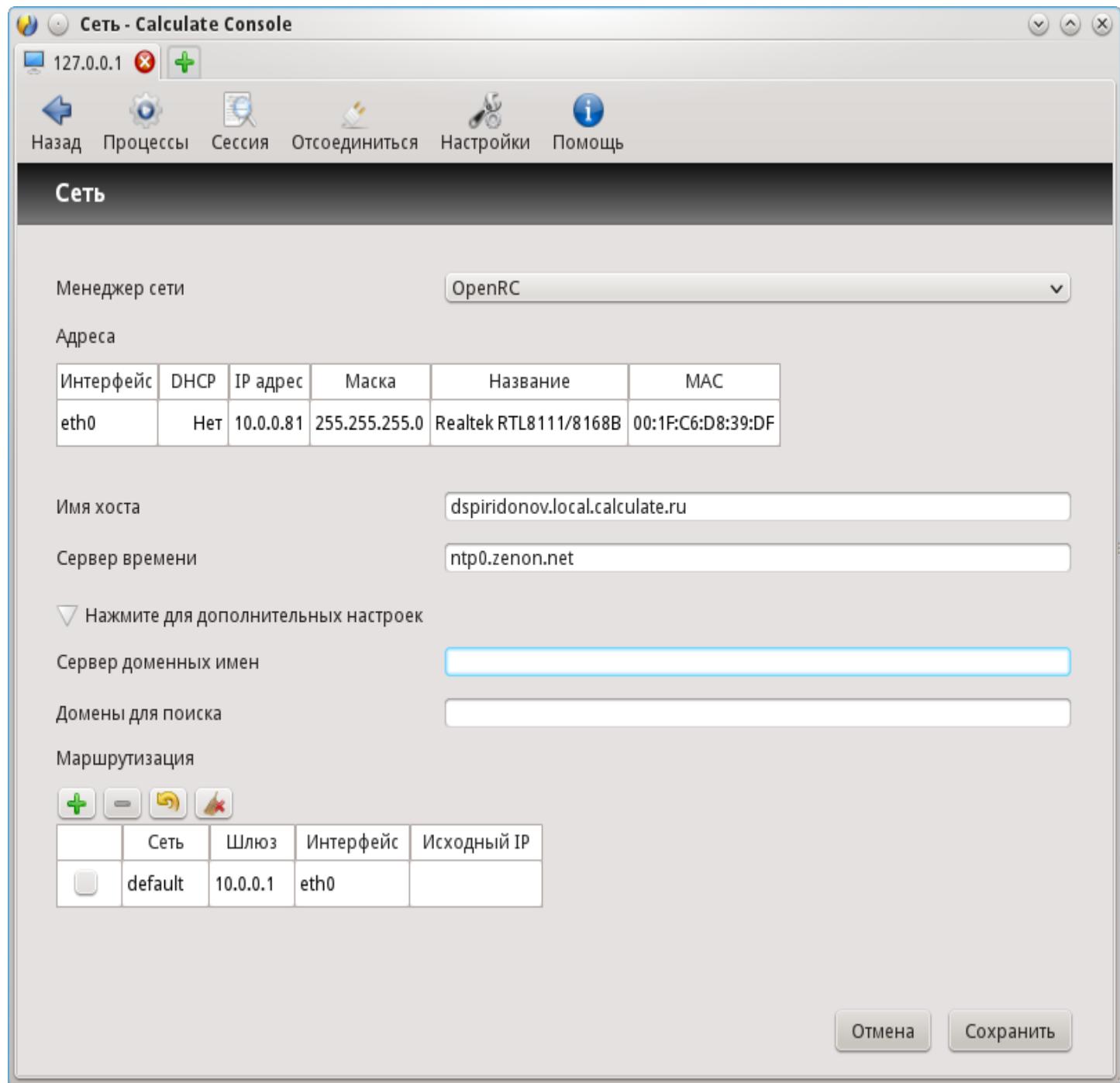
```
указываем нужный MAC-адрес
mac_eth0="01:23:45:67:89:AB"
```

## Настройка сети

- [Настройка сети](#)
- [Настройка графическим клиентом](#)
- [Настройка консольным клиентом](#)
- [Настройка сервером утилит](#)
- [Настройка без использования утилит Calculate](#) Настройка сети включает в себя основные параметры:
  - выбор менеджера сети;
  - установка адреса для интерфейса (DHCP, IP адрес и маска);
  - установка имени хоста;
  - установка сервера времени; и дополнительные параметры:
    - установка сервера доменных имён;
    - установка доменов для поиска;
    - установка маршрутизации (сеть, шлюз, интерфейс, исходный IP).

### Настройка графическим клиентом

Настройка сети для графического клиента *cl-console-gui* находится в категории **Настройка**. Для настройки локализации необходимо выбрать требуемые параметры, затем нажать кнопку **Сохранить**. Дополнительные параметры можно открыть нажав на соответствующую кнопку.



## Настройка консольным клиентом

Для настройки сети с помощью консольного клиента cl-console используйте команду  
cl-console --method setup\_network

Назначение ключей:

- --netconf NETMANAGER - выбор менеджера сети (networkmanager или openrc)
- --iface IFACE\_SETTINGS - установка адреса для сетевого интерфейса
- --hostname HOSTNAME - установка короткого или полного имени хоста
- --ntp NTP - установка NTP сервера для системы
- --dns DNS - установка серверов доменных имен (запятая - разделитель)
- --domain-search DOMAINS - установка доменов для поиска (запятая - разделитель)
- --route NETROUTE - добавить правило маршрутизации

Пример:

```
cl-console --method setup_network --netconf openrc --iface
eth0:192.168.1.47:24 --hostname iivanov.company.ru --route
default:192.168.1.1:eth0:192.168.1.47
```

## Настройка сервером утилит

Для настройки с помощью сервера утилит используйте команду  
`cl-setup-network`

Все ключи и их назначение совпадают с ключами, описанными выше в разделе настройка консольным клиентом.

Пример:

```
cl-setup-network --netconf openrc --iface eth0:192.168.1.47:24 --hostname
iivanov.company.ru --route default:192.168.1.1:eth0:192.168.1.47
```

## Настройка без использования утилит Calculate

Настройку сети в CLD, CLDG и CLDX выполняет *Networkmanager*. Программа имеет графический интерфейс, при помощи которого можно настроить как проводную, так и беспроводную сеть. При загрузке программа попытается настроить сетевой интерфейс и получить IP-адрес от DHCP-сервера. Доступ к программе можно получить, кликнув по иконке в правом верхнем углу экрана.

Настройка сети в CLS, CDS и CSS осуществляется путем редактирования конфигурационного файла `/etc/conf.d/net`. Справку по настройке сети можно найти в файле `/usr/share/doc/openrc-0.6.x/net.example`. После изменения настроек нужно перезапустить сетевой интерфейс (как правило, `eth0` для проводной сети):

```
/etc/init.d/net.eth0 restart
```

# 8. Настройка оборудования

1. [Настройка звука](#)
2. [Настройка софтового модема](#)
3. [Настройка TV тюнера AVermedia AVerTV 305/307](#)
4. [Настройка сканера Epson Perfection 1670](#)
5. [Настройка Wake-on-Lan](#)

## Настройка звука

За вывод звука в Calculate Linux отвечает звуковой драйвер ALSA. Во время первой загрузки, установщик Calculate Linux устанавливает громкость на каналах в 70%. Тем не менее в некоторых случаях звук в динамиках может отсутствовать. Для обнаружения вашей звуковой карты воспользуйтесь консольной утилитой `alsamixer`.

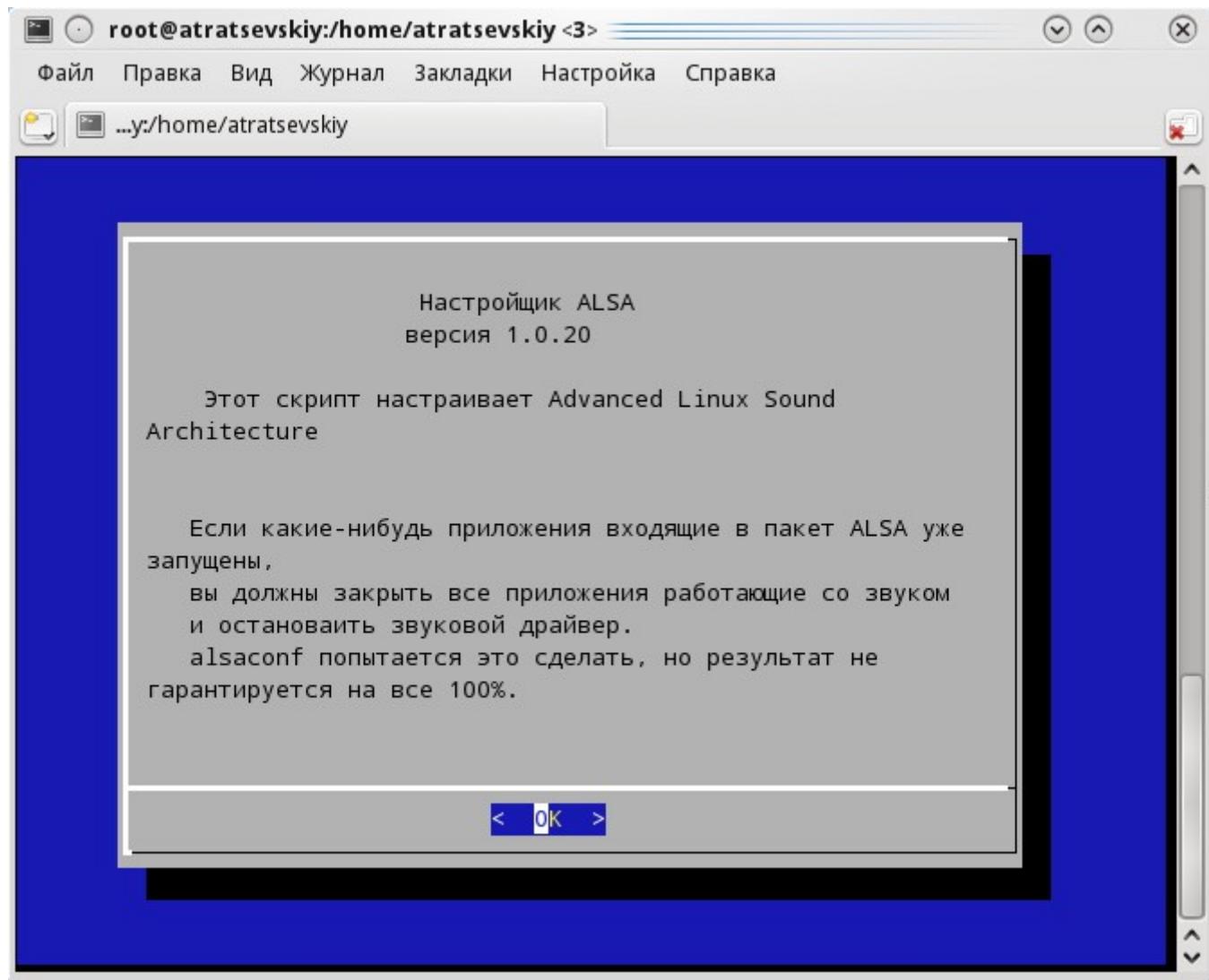


Рис. 1. Окно приветствия программы Alsacnf.

Далее нажмите OK, чтобы перейти к диалоговому окну выбора звуковой карты.

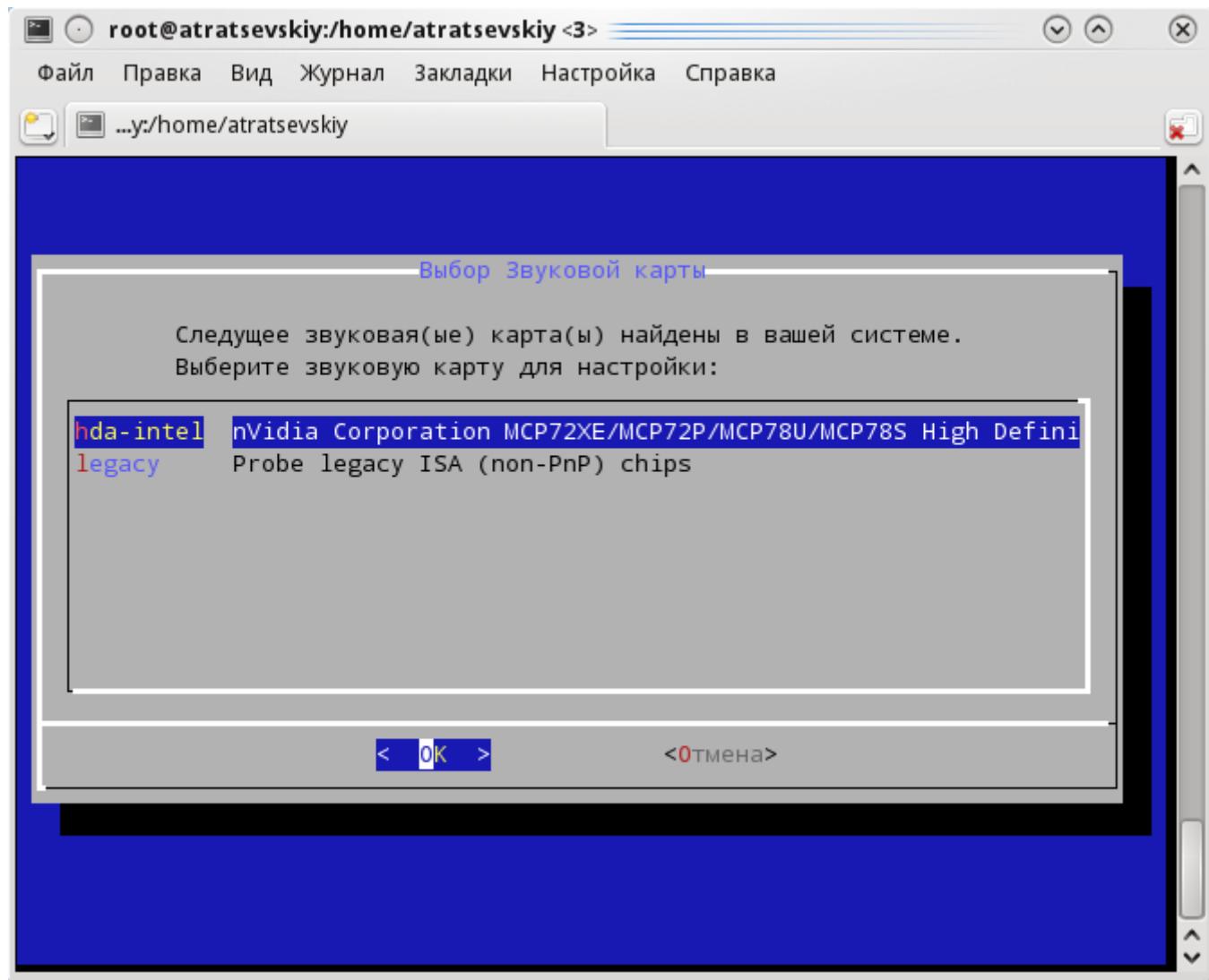


Рис. 2. Выбор звуковой карты.

Выберите вашу карту и нажмите ОК. Последующий диалог попросит подтвердить ваш выбор, после чего настройки звуковой карты будут сохранены.

Управлять громкостью на каналах можно как при помощи утилит, входящих в оконные менеджеры KDE, XFCE, Gnome и др., так и при помощи консольной утилиты alsamixer.

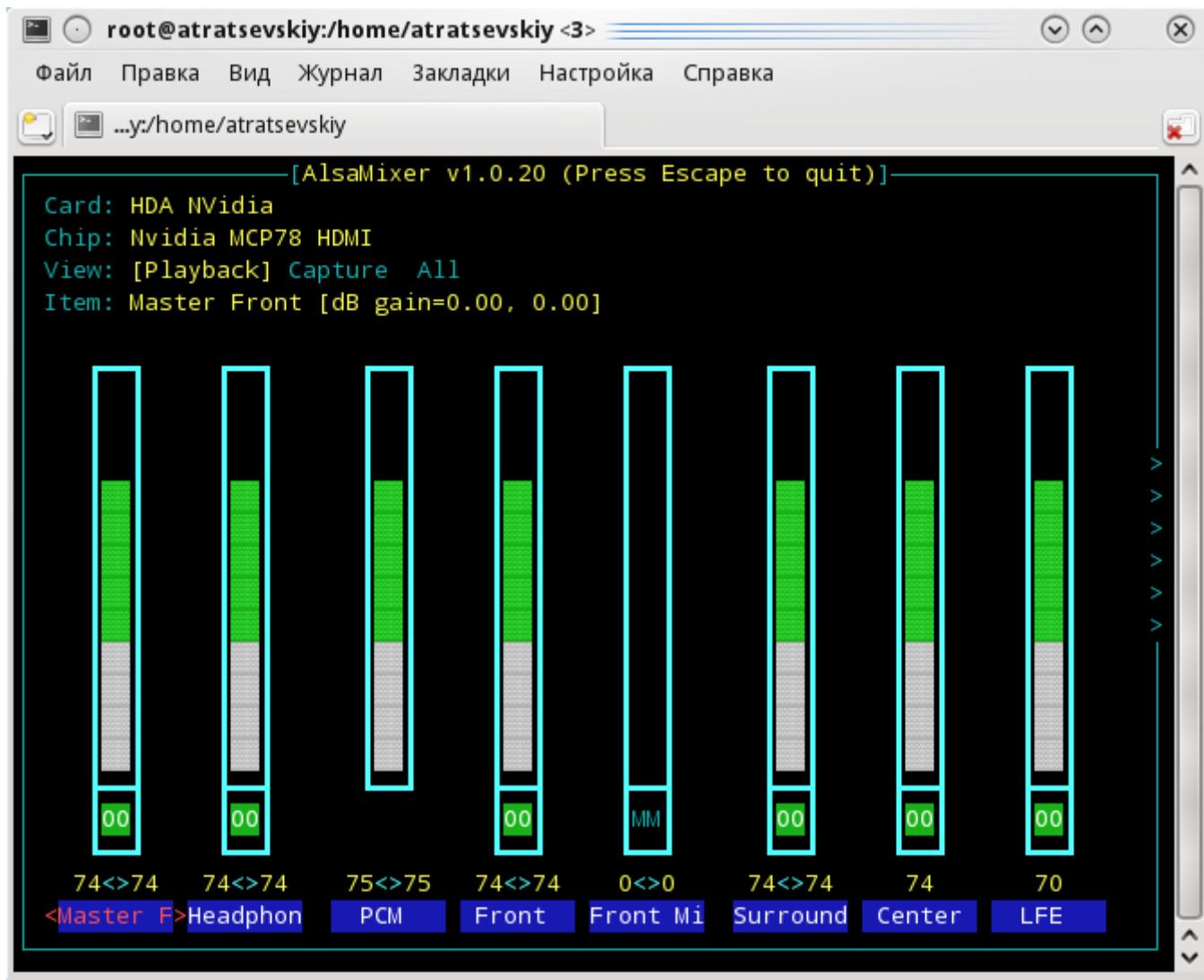


Рис. 3. Вывод окна Alsamixer.

Проверить правильность подключения ваших колонок можно при помощи утилиты `speaker-test`.

Для звука "5.1" выполните в консоли команду:

```
speaker-test -Dplug:surround51 -c6
```

## Настройка софтового модема

### Поддержка

Большинство софтовых модемов поддерживается в [Calculate Linux](#). Программа установки не определяет наличие в компьютере модема, его нужно активировать самостоятельно.

### Настройка

Для запуска демона модема выполните:

```
/etc/init.d/slmodem start
```

Если программа успешно определит модем, вы найдете новое устройство `/dev/ttySL0`. В противном случае можно попробовать изменить параметр `_HW_SLOT` файла `/etc/conf.d/slmodem`, изменив его значение на "modem:1", и выполнить:

```
/etc/init.d/slmodem restart
```

## Автозапуск демона

Для добавления демона в автозагрузку выполните:

```
/sbin/rc-update add slmodem default
```

# Настройка TV тюнера Avermedia AVerTV 305/307

- [Настройка TV тюнера Avermedia AVerTV 305/307](#)
- [Проверка наличия необходимых модулей](#)
- [Загрузка модулей](#)
- [tvtime](#)
- [Телевизионный стандарт SECAM](#)
- [Телевизионный стандарт PAL](#)
- [Настройка просмотра для двух стандартов PAL,SECAM](#)
- [mplayer](#)
- [Просмотр](#)
- [Запись](#)

## Проверка наличия необходимых модулей

Необходимо убедиться, что в системе присутствуют следующие модули: "tuner", "saa7134", "tda9887".

Проверяем наличие модулей:

```
modprobe tuner
modprobe saa7134
modprobe tda9887
```

## Загрузка модулей

В каталоге /etc/modules.d необходимо создать любой файл, например saa7134:

```
touch /etc/modules.d/saa7134
```

Содержимое файла saa7134:

```
alias char-major-81 videodev
alias char-major-81-0 saa7134
options saa7134 card=102 tuner=38 secam=dk i2c_scan=1 alsal=1
options tuner secam=d
options tda9887 port2=0 port1=1
```

Для загрузки модулей при запуске системы нужно выполнить:

```
update-modules --force
```

Перезагружаемся.

## tvtime

Устанавливаем tvtime:

```
emerge media-tv/tvtime
```

## Телевизионный стандарт SECAM

### Первый вариант настройки

Выбираем tv-стандарт SECAM, таблицу каналов RUSSIA и выполняем поиск (команды в меню *tvttime*).

### Второй вариант настройки

1. Выбираем tv-стандарт SECAM, таблицу каналов CUSTOM.
2. Выходим из *tvttime*.
3. Выполняем:

```
tvttime-scanner
tvttime
```

## Телевизионный стандарт PAL

Аналогично SECAM, только меняем стандарт на PAL.

## Настройка просмотра для двух стандартов PAL,SECAM

### Первый вариант

1. Выбираем tv стандарт SECAM, таблицу каналов RUSSIA и выполняем поиск.
2. Выбираем tv стандарт PAL, рестартуем выполняем поиск.
3. Редактируем файл `~/.tvttime/stationlist.xml`, там будет 2 раздела.  
Для режима SECAM перечислены все каналы и для каждого своя строка с настройками.  
Параметр `active` говорит о том, что канал настроен (есть сигнал - 1, нет сигнала - 0).  
Ниже - тоже самое для режима PAL с его активными каналами.  
Для каждого канала есть параметр `norm=SECAM` или `PAL` соответственно.
4. В раздел SECAM добавляем, где нужно, каналы из раздела PAL с `active=1, norm=PAL`.
5. Выбираем tv-стандарт SECAM, рестартуем.

### Второй вариант

Выполнить:

```
tvttime-scanner -n PAL && tvttime-scanner -n SECAM
```

Отредактировать файл `~/.tvttime/stationlist.xml` аналогично первому варианту (подразделы с 3 по 5)

## mplayer

### Просмотр

```
mplayer tv:// -tv device=/dev/video0:driver=v4l2:freq=144.25:normid=17
```

где: \* freq=144.25 - частота; можно посмотреть после сканирования в `~/.tvttime/stationlist.xml`  
\* normid=17 - SECAM, телевизионный стандарт (при запуске *mplayer* показывает доступные стандарты и номера normid)

### Запись

```
mencoder tv:// -tv device=/dev/video0:driver=v4l2:freq=144.25:normid=17
-ffourcc XVID -ovc lavc -lavcopts vcodec=mpeg4:vbitrate=1800:v4mv -vf
pp=fd -audio-demuxer 20 -oac mp3lame -lameopts cbr:preset=128:mode=1
-o ./Video.avi
```

ffourcc XVID - совместимость с бытовыми плеерами и windows

## Видеокодек *mpeg4*

Битрейт видео *vbitrate=1800*. Чем больше, тем качественней запись (больше загрузка процессора).

Значение больше 2500 имеет смысл только для очень качественного оригинала.

Бытовые плееры поддерживают примерно до 2000.

## Удаление гребенки *-vf pp=fd*

Кодек звука mp3 (должен быть установлен аудиокодек *lame*)

Битрейт звука 128

Пишет в директорию, из которой запущен, файл *./Video.avi*.

# Настройка сканера Epson Perfection 1670

## Установка SANE

Для работы сканеров в GNU\Linux необходим API, разрабатываемый проектом [SANE](#). На сайте проекта находится страница со всеми [поддерживаемыми устройствами](#); там же описано качество работы конкретных устройств, указан backend, осуществляющий поддержку. Поддержка нашей модели осуществляется backend'ом *snapscan*.

В [Calculate Linux Desktop](#) SANE и XSANE (графический интерфейс к SANE) установлены по умолчанию, но если в вашей системе данные программы отсутствуют, воспользуйтесь следующими командами:

```
emerge sane-backends xsane
```

(пользователям kde4 может пригодиться пакет *libksane*, предоставляющий интеграцию среды с интерфейсом sane)

## Определение сканера

Удостоверимся, что сканер определился системой. Воспользуемся командой

```
lsusb
```

(потребуются права администратора)

Ответом на данную команду должен быть список устройств, подключенных по usb, среди которых должна быть строка с названием нашего сканера:

```
Bus 001 Device 002: ID 04b8:011f Seiko Epson Corp. Perfection 1670
```

В противном случае смотрим, что нам скажет команда:

```
dmesg | grep error
```

Если обнаруживаются строки вроде этих:

```
usb 2-9: device descriptor read/64, error -71
usb 2-9: device not accepting address 9, error -71
```

- проверьте кабель. Некоторые устройства требуют подключения к контроллеру USB 2.0 через скоростной кабель.

Также воспользуемся утилитой *sane-find-scanner*, которая должна вывести список опознанных сканеров. В выводе мы должны увидеть строку:

```
found USB scanner (vendor=0x04b8 [EPSON], product=0x011f [EPSON Scanner])
at libusb:001:002
```

Естественно шина usb (libusb:001:002) у вас может быть другая.

## Установка прошивки

Определенные модели не смогут работать без фирменного программного обеспечения (прошивки). Нам также потребуется прошивка. Прошивки не включаются в дистрибутив SANE из-за лицензионных соглашений, но в большинстве случаев доступны на сайте производителя или находятся на диске, который идет в комплекте со сканером. Иными словами, нам понадобятся драйверы для windows. Я скачивал драйверы с [официального сайта](#). В архиве с драйверами находим файл ModUsb.cab - здесь нам понадобится утилита cabextract:

```
emerge cabextract
```

Переходим в каталог с драйверами и выполняем команду

```
cabextract ModUsb.cab
```

Среди распакованных файлов находим esfw30.bin - это и есть наша прошивка.

## Настройка SANE

Далее скопируем файл прошивки в каталог /usr/share/sane/snapscan/ и отредактируем файл конфигурации необходимого нам backend'a. Откроем файл /etc/sane.d/snapscan.conf и поправим строку с адресом прошивки:

```
firmware /usr/share/sane/snapscan/Esfw30.bin
```

Затем выключим сканер, отключим от него usb-кабель, заново подключим к usb и включим питание сканера. Настало время запустить XSANE, для начала от пользователя root. Если у вас установлен только сканер и нет другого оборудования (тюнер или вебкамера), то перед вами при запуске xsane предстанет диалог выбора устройства. Нас интересует наш сканер, так что выбираем "EPSON Scanner1" и нажимаем "OK". Немного подумав, должна запуститься xsane. Если этого не произошло, то, вероятней всего, вы увидите окошко с ошибкой. Сообщение "Invalid argument" означает что у нас что-то с прошивкой. Текст ошибки, содержащий "I/O", скорее всего означает, что у нас проблемы с обращением к устройству. Если же всё нормально, открывается окно программы сканирования - можно проверить сканер в действии!

Но работа от пользователя root - не лучший способ, поэтому нам нужно добавить пользователей, которым необходим доступ к сканеру, в группу scanner:

```
gpasswd -a user1,user2 scanner
```

## Настройка Wake-on-Lan

Wake-On-Lan - технология, позволяющая включать компьютер по сети.

### Требование к ведомому компьютеру

- ATX источник питания, материнская плата с поддержкой Wake-On-Lan
- сетевой адаптер с поддержкой Wake-On-Lan
- известный MAC-адрес сетевого адаптера

### Требование к ведущему компьютеру

- специальная программа, умеющая отсылать Magic Packet

## Принцип работы

Ведомый компьютер находится в дежурном режиме (*stand by*) и выдает питание на сетевой адаптер. Сетевой адаптер находится в режиме пониженного энергопотребления, просматривая все пакеты, приходящие на его MAC-адрес, но ничего не отвечая на них. Если один из них окажется Magic Packet, то сетевой адаптер выдаёт сигнал на включение питания компьютера.

## Реализация

Включаем поддержку *WoL* в *BIOS* на ведомом компьютере. Это может быть одноименный пункт наподобие *Wake On Lan Enable*, либо *Power On By PCIE* и т.д., может также быть, что этот режим в *BIOS* не меняется, а материнская плата поддерживает его по умолчанию.

Чтобы определить, поддерживает ли сетевая карта *WoL*, - загружаем ведомый компьютер набираем в консоли

```
ethtool eth0
```

Получаем результат:

```
Settings for eth0:
```

```
 Supported ports: [MII]
 Supported link modes: 10baseT/Half 10baseT/Full
 100baseT/Half 100baseT/Full
 1000baseT/Full
 Supports auto-negotiation: Yes
 Advertised link modes: 10baseT/Half 10baseT/Full
 100baseT/Half 100baseT/Full
 1000baseT/Full
 Advertised auto-negotiation: Yes
 Speed: 1000Mb/s
 Duplex: Full
 Port: MII
 PHYAD: 1
 Transceiver: external
 Auto-negotiation: on
 Supports Wake-on: g
 Wake-on: d
 Link detected: yes
```

Нас интересуют строчки *Supports Wake-on* и *Wake-on*. Первая показывает доступные режимы сетевого адаптера на пробуждение (*g* - как раз пробуждение по *Magic Pocket*), а вторая - текущий режим (*d* означает выключенный *WoL*).

Для того, чтобы перевести сетевую карту в режим *WoL*, используется команда

```
ethtool -s eth0 wol g
```

Для выключения режима *WoL*

```
ethtool -s eth0 wol d
```

Сетевой адаптер может поддерживать сохранение состояния, в которое его перевели, но может и сбрасывать (чаще всего на *d*), поэтому при каждой загрузке необходимо будет устанавливать нужный режим *WoL*. Добавляем в */etc/conf.d/net* следующие строки - они будут включать режим *WoL* на всех сетевых адаптерах, которые его поддерживают:

```
preup() {
 if ethtool $1 | grep "Supports Wake-on:" | grep g >/dev/null;
 then
 ethtool -s $1 wol g
 fi
}
```

Для получения MAC-адреса сетевого адаптера на ведомом компьютере можно

- выполнить команду на ведомом компьютере

```
ifconfig -a
eth0 Link encap:Ethernet HWaddr 01:02:03:04:05:06
 inet addr:192.168.1.2 Bcast:192.168.1.255 Mask:255.255.255.0
 inet6 addr: fe80::215:f2ff:fe6f:3487/64 Scope:Link
 UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
 RX packets:71495 errors:0 dropped:0 overruns:0 frame:0
 TX packets:76190 errors:0 dropped:0 overruns:0 carrier:0
 collisions:0 txqueuelen:1000
 RX bytes:23164212 (22.0 MiB) TX bytes:7625016 (7.2 MiB)
 Interrupt:217 Base address:0xd400
```

- выполнить команду на ведущем компьютере, которая отобразит ARP-кэш

arp

Address	HWtype	HWaddress	Flags	Mask
Iface				
10.0.0.1	ether	00:01:02:03:04:05	C	
eth0				
10.0.0.2	ether	06:07:08:09:0a:0b	C	
eth0				
10.0.0.3	ether	0c:0d:0e:0f:10:11	C	
eth0				

- для того, чтобы все компьютеры сети попали в кэш, можно воспользоваться утилитой *nmap*, которая пропингует компьютеры в сети и их MAC-адреса попадут в кэш

*nmap -v -Sp 10.0.0.0/24*

Для пробуждения компьютера используем утилиту *wol* на ведущем компьютере

*wol* MAC-адрес

При работе с программами следует учитывать, что не все компьютеры включаются сразу после подключения в электрическую сеть. Это связано с отсутствием процесса инициализации подачи питания на сетевую карту (компьютер еще не включался и не знает, какие устройства следует питать чтобы получать специальные сигналы, среди которых будет магический пакет). Поэтому следует произвести одно предварительное включение вручную. Если существует необходимость избавиться от данной проблемы (например, сервер закрывается на ключ или находится очень далеко), следует установить в BIOS параметр питания *Wake After Power Fail* в значение *ON*.

## 9. Справка по основным командам Gentoo/Calculate

- [portage](#) - система управления пакетами в Gentoo
- [eix](#) - набор утилит для поиска eбилдов по дереву Portage и получения информации о них, в том числе работа с локальными настройками, внешними оверлеями, версиями пакетов и др.
- [layman](#) - утилита для управления оверлеями Gentoo
- [openrc](#) - система управления службами системы, запуском и завершением работы хоста
- [portage-utils](#) - набор легких и быстрых утилит, написанных на C, для извлечения информации о пакетах в Portage
- [gentoolkit](#) - набор скриптов для администрирования систем, работающих на Gentoo
- [gentoolkit-dev](#) - набор скриптов в помощь разработчикам под Gentoo

## НАЗВАНИЕ

**eix** - набор утилит для поиска, определения различий и обновления бинарного кэша из вашего локального дерева

## СИНТАКСИС

**eix** [*общие опции*] [*ОПЦИИ*] **ВЫРАЖЕНИЕ**

**eix-update** [*общие опции*] [*опции eix-update*]

**eix-diff** [*общие опции*] **СТАРЫЙ КЭШ** [**НОВЫЙ КЭШ**]

**eix-sync**

**eix-test-obsolete**

**eix-remote**

**eix-layman**

**eix-installed-after**

**eix-installed**

**eix-functions.sh**

**versionsort**

## ОПИСАНИЕ

**eix-update** генерирует бинарный кэш локального дерева портежей и оверлеев. **eix** позволяет осуществлять в этом кэше поиск, ограничивая его условиями, указанными в **ВЫРАЖЕНИИ**. Соответственно, если вы не указываете таких условий, будут выведены все пакеты. **eix-diff** сравнивает два бинарных кэша на предмет обнаружения пакетов, которые были добавлены, удалены или пакеты с новыми стабильными версиями.

Все перечисленные программы и сценарии обращаются к конфигурационным файлам, которые будут описаны ниже. **eix-sync** дополнительно имеет собственный конфигурационный файл.

**eix-sync** умеет синхронизировать дерево портежей/оверлеев и сравнивать новые данные со старым кэшем с помощью **eix-diff**. Для получения подробной справки по **eix-sync** выполните **eix-sync -h**. Вам также следует обратиться к документации по файлу */etc/eix-sync.conf* (см. ниже); заметьте, что его содержимое может также сохраняться в переменной **EIX\_SYNC\_CONF**.

**eix-test-obsolete** - сценарий, несколько раз вызывающий **eix** для структурированного вывода **eix -tTc**.

**eix-remote** позволяет синхронизировать текущую базу **eix** с внешним сервером, добавляя/удаляя из неё данные. Заметьте, что при следующем вызове **eix-update** данные, полученные с сервера, обнулятся; чтобы обойти это поведение, следует включить **KEEP\_VIRTUALS=true** в */etc/eixrc*. Для получения подробной справки по **eix-remote** выполните **eix-remote -h**, а также обратитесь к документации по файлу */etc/eix-remote.conf* (см. ниже).

**eix-layman** умеет добавлять локальные оверлеи **layman** к текущей базе данных и удалять их из нее. Этот сценарий полезен в том случае, если вы не указываете источник данных для **layman** в локальном */etc/make.conf*. Для получения подробной справки по **eix-layman** выполните **eix-layman -h**. **eix-layman** также является примером использования **eix-functions.sh**.

**eix-installed-after** - простой и хорошо откомментированный сценарий, демонстрирующий некоторые возможности кастомизации формата вывода **eix**. Он выводит перечень пакетов, установленных до (или после) последней (или первой) установки определённого пакета/версии. Для получения подробной справки выполните **eix-installed-after -h**.

**eix-functions.sh** предоставляет вспомогательные функции, вызываемые **eix-sync**, **eix-remote** и **eix-layman**. Их можно использовать и при написании собственных подобных сценариев. Заметьте, что сразу после назначения источников данных для **eix-functions.sh**, вероятно, потребуется вызвать **read\_functions [АРГУМЕНТЫ]** для передачи соответствующих АРГУМЕНТОВ **functions.sh** программе **baselayout**.

**eix-installed** представляет собой простой сценарий, который выводит все установленные пакеты (с их точными версиями), а также может осуществлять поиск пакетов, установленных как с информацией о репозитарии или данными времени сборки, так и без таковой (ср. описание переменных **CHECK\_INSTALLED\_OVERLAYS** и **USE\_BUILD\_TIME**). Для получения подробной справки по **eix-installed** выполните **eix-installed -h**.

**versionsort** - вспомогательная утилита для сценариев, которая убирает номера версий из аргументов и выводит их, отсортировав согласно правилам сортировки версий portage. Подробнее см. в конце этой man-страницы.

## ПРИМЕРЫ

Нижеследующие примеры демонстрируют некоторые полезные, но не вполне привычные случаи использования eix. Мы приводим их, чтобы показать широту возможностей утилиты. Поскольку предполагается, что примеры могут копироваться непосредственно со справочной страницы, мы приводим их здесь с минимальным описанием. Для понимания принципа их работы следует, разумеется, полностью ознакомиться с руководством. Дополнительные примеры можно посмотреть, например, в сценарии **eix-installed-after** - он снабжен пространными комментариями.

**команда| eix '-\*! --format "**

р(((. Подразумевая, что команда создает список в формате **категория/пакет-версия** или **=категория/пакет-версия**, будет выведен соответствующий список в виде **категория/пакет** или **категория/пакет:СЛОТ** (в зависимости от того, релевантно ли значение СЛОТА).

**команда| eix '-\*! --format "**

Аналогично предыдущей команде, но вывод всегда будет в виде **категория/пакет:СЛОТ**, даже если значение СЛОТА излишне. (Легко запомнить: Always ("всегда")).

**eix '-I\*! --format "**

Будут отображены установленные пакеты в виде **категория/пакет-версия**. Очевидно, что формат вывода можно изменить, вместо **NAMEVERSION** используя **NAMESLOT** или **NAMEASLOT**. В таком окружении единственное предназначение опции **-I** заключается в некотором ускорении вывода.

**eix '-I\*! --format "**

Аналогично предыдущей команде, но вывод будет произведен в виде **=категория/пакет-имя**, обеспечивая тем самым прямую передачу данных portage.

**eix '-I\*! --format " | sort -n | cut -f2-3**

Будут выведены установленные пакеты (если слоты имеют значение, то со слотами), в порядке, соответствующем дате установки. Е сли вам желателен вывод в ином формате, следует заменить **DATESORT** подходящей переменной (исходное определение вывода вы можете видеть, выполнив **eix --dump**).

Вот как определяется порядок вывода запрошенных пакетов: переменная **DATESORT** формирует первый столбец, отображая в нем прошедшее время в секундах (если выполнить команду **eix --print DATESORT**, вы увидите, что за это отвечает переменная **DATESORT\_DATE**, первая запись в которой - **%s**). Затем программа **sort** перестраивает список пакетов в алфавитном порядке. В последнюю очередь вызывается программа **cut**, которая отсекает первый столбец, который был нужен только для сортировки.

В приведенных выше примерах **NAMEVERSION** и **DATESORT** - имена переменных, предопределенных в eix (чтобы их увидеть, выполните **eix --dump**). Но ничто не мешает вам также определить и использовать собственные переменные. Работа с ними описана в man-руководстве; обратите особое внимание на описание строки **FORMAT**. }}}

# ОПЦИИ

## Общие опции

Здесь перечисляются опции, общие для **eix**, **eix-diff** и **eix-update**.

### **-h, --help**

Вывести справку и выйти.

### **-Q, --quick** (переключатель) (не используется с **eix-update**)

(Не) считывать слоты установленных версий, которые невозможno предположить (например, установленные версии пакетов с как минимум двумя различными слотами, для которых в базе данных уже нет установленной версии). Учтите, что при использовании этой опции **eix** и **eix-diff** могут выдавать ложноположительные результаты при рекомендациях по обновлению/откату таких пакетов.

### **--care** (не используется с **eix-update**)

Отключить опцию **--quick**; кроме того, слоты установленной версии всегда будут считываться, а не определяться предположительно. В частности, при изменении имени слота установленной версии система проверит наличие рекомендации по обновлению/откату. Учтите, что при этом значительно замедлится скорость первого вызова. (Если ваша файловая система использует кэш разумной величины, только первого - последующие вызовы должны выполняться почти с такой же скоростью, как если бы эта опция не использовалась.)

### **-q, --quiet** (переключатель)

Отключить стандартный вывод утилиты в консоль. Вы можете уменьшить время выполнения **eix**, используя эту опцию в связке либо с **--brief**, либо с **--brief2** (в зависимости от того, что вам нужно) и установив значение переменной **COUNT\_ONLY\_PRINTED=false**. См. также переменные **NOFOUND\_STATUS** и **MOREFOUND\_STATUS**

### **--dump**

Показать действующие значения переменных **eixrc**, а значения по умолчанию будут отображены как комментарии; затем выйти.

### **--dump-defaults**

Показать значения по умолчанию переменных **eixrc**, а действующие значения будут отображены как комментарии; затем выйти.

### **--print ПЕРЕМЕННАЯ**

Вывести значение указанной аргументом **ПЕРЕМЕННОЙ** **eixrc** или **portage**, в расширенном виде, предназначенном для внутреннего использования **eix**; затем выйти. Прежде всего это полезно в сценариях или для отладки. Если вы используете данную опцию в сценариях, вам может потребоваться указать также значение **PRINT\_APPEND** для поддержки закрывающих пробелов (см. описание **PRINT\_APPEND**).

### **-V, --version**

Вывести номер версии и выйти.

### **-n, --nocolor**

Отключить использование цветового кода ANSI. Это имеет смысл для терминалов, не поддерживающих ANSI. (Данная опция автоматически включается, если стандартный вывод - не терминал, но может быть переопределена использованием опции **--force-color**)

### **-F, --force-color**

Поведение, обратное **--nocolor**.

## **Особые опции информирования**

Следующие особые опции информирования работают только с бинарным кэшем **eix**. Это одноразовые исключающие опции - иными словами, при их использовании **eix** выведет только требуемые данные и завершит работу.

### **--print-overlay-path** оверлей\_

Вывести путь к первому оверлею, соответствующему запрошенному **ОВЕРЛЕЮ**. В качестве **ОВЕРЛЕЯ** можно указывать метку оверлея, путь (маску) или число.

### **--print-overlay-label** оверлей

Вывести метку оверлея, соответствующую запрошенному **ОВЕРЛЕЮ**. В качестве **ОВЕРЛЕЯ** можно указывать метку оверлея, путь (маску) или число.

### **--print-all-useflags**

Вывести все IUSE words, используемые в определенном пакете.

### **--print-all-keywords**

Вывести все ключевые слова, используемые в определенном пакете.

### **--print-all-slots**

Вывести все строки слотов, используемые в определенном пакете.

### **--print-all-provides**

Вывести все строки PROVIDE, используемые в определенном пакете.

### **--print-all-licenses**

Вывести все строки LICENSE, используемые в определенном пакете.

### **--print-world-sets**

Вывести сеты world.

### **--is-current**

Проверить только, доступен и корректен ли **/var/cache/eix** (в версии, поддерживаемой бинарным **eix**). Если да, то **eix** успешно завершает работу, не выводя никаких сведений.

## **Опции вывода**

### **-x, --versionsort** (переключатель)

Вывести доступные версии, отсортировав их по слотам/версиям. Если сортировка производится по слотам, на каждой строке должен располагаться только один слот.

### **-l, --versionlines** (переключатель)

Вывести доступные версии (вертикальным списком). Кроме того, данные о IUSE будут отображаться отдельно для каждой версии (а не для всего пакета).

### **-c, --compact** (переключатель)

Использовать компактный формат вывода результатов поиска **eix**. Это удобно, если выводится длинный перечень; кроме того, тем самым ускоряется поиск на медленных соединениях, например, при использовании серийной консоли.

### **-v, --verbose** (переключатель)

Использовать подробный вывод с дополнительной информацией о результатах поиска - например, сообщать о лицензии, под которой распространяется пакет.

### **--xml** (переключатель)

Использовать вывод в формате XML. Если вы хотите передать эту опцию внешней программе, возможно, потребуется использовать ее в связке с **--care** и экспортировать некоторые переменные, такие как **LOCAL\_PORTAGE\_CONFIG**, чтобы убедиться, что пользовательские настройки вывода не влияли на ваш вывод. Когда опция включена, автоматически включаются также **OVERLAYS\_LIST=none** и **--pure-packages**. Формат вывода можно незначительно модифицировать с помощью переменных **XML\_\***. Используемый XML-формат документирован и хранится в удобочитаемом виде в файле eix-xml.html или eix-xml.txt, а в менее удобном для восприятия виде (как xml-схема) - в файле eix-xml.xsd.

#### **-\*, --pure-packages** (переключатель)

(Не забывайте ставить кавычки, если используете краткую форму в оболочке.) Исключить вывод в конце дополнительной информации (имен оверлеев, количество найденных пакетов). Это может пригодиться при использовании сценариев интерпретатора с парсингом вывода.

#### **--only-names** (переключатель)

Как **"-\*"**, но дополнительно выводить только категории и имена запрошенных пакетов.

#### **-0, --brief** (переключатель)

Вывести максимум один пакет и остановиться. Как правило, выполнение этой опции можно ускорить, назначив **COUNT\_ONLY\_PRINTED=false**. В этом последнем случае при поиске по приблизительному соответствию могут быть выведены не самые последние доступные для установки версии.

#### **--brief2** (переключатель)

Как **--brief**, но вывод будет ограничен двумя пакетами.

## **Особые опции eix**

#### **-t, --test-non-matching**

Перед очередным выводом отображать записи в `/etc/portage/package.*`, которые не соответствуют ни одной существующей версии в базе данных пакетов или пусты и не имеют смысла (см. **TEST\_FOR\_EMPTY**).

Эта опция выводит также все установленные, но не фигурирующие в базе данных пакеты.

Заметьте, что она действует совершенно иначе, нежели **-T** (см. ниже), которая только проверяет пакеты в базе данных на предмет наличия в `/etc/portage/package.*` дублированных записей или, соответственно, доступности установленных версий.

Эту опцию предпочтительнее использовать в связке с **-T**, для очистки `/etc/portage/package.*`

Можно совмещать ее и с опцией **-e**, чтобы избежать дополнительного вывода.

Если по какой-либо причине вы хотите исключить отдельные записи/пакеты из числа проверяемых, вам следует поместить соответствующие записи в файл `/etc/portage/package.*.nonexistent`, где `*=keywords,mask,unmask,use,env,cflags,installed`. Эти файлы и порядок их переименования будут описаны ниже.

#### **--cache-file** *ФАЙЛ*

Использовать *ФАЙЛ* вместо `/var/cache/eix`.

## **Опции ВЫРАЖЕНИЯ**

ВЫРАЖЕНИЕ используется для ограничения вывода пакетов при вызове eix.

ВЫРАЖЕНИЕ может содержать булевые операторы и условия, согласно следующему синтаксису:

ВЫРАЖЕНИЕ ::= [ **--not** | **!** ] ФИГУРНАЯ\_СКОБКА\_ИЛИ\_ПРОВЕРКА |

ВЫРАЖЕНИЕ [ **--and** | **-a** ] ВЫРАЖЕНИЕ |

ВЫРАЖЕНИЕ [ **--or** | **-o** ] ВЫРАЖЕНИЕ |

ФИГУРНАЯ\_СКОБКА\_ИЛИ\_ПРОВЕРКА ::= **--open** | -( ВЫРАЖЕНИЕ **--close** | - ) |

ПРОВЕРКА\_ПО\_КРИТЕРИЯМ

## ПРОВЕРКА\_ПО\_КРИТЕРИЯМ ::= [КРИТЕРИИ\_ПРОВЕРКИ] [ШАБЛОН]

Не забывайте, что в интерпретаторе символы **!**, **(**, **)** необходимо заключать в кавычки, чтобы eix распознал их как часть аргумента!

Если вам необходимо, чтобы ВЫРАЖЕНИЕ начиналось с **-**, поставьте впереди два дефиса подряд: **--**. Тогда ВЫРАЖЕНИЕ не будет воспринято как опция, а добавочные символы **--** будут проигнорированы. Например, команда **eix ---tool --or ---util** выведет пакеты, содержащие **-tool** или **-util**.

Думается, значение логических операторов очевидно - за исключением, может быть, следующих особенностей:

1. Если между двумя ВЫРАЖЕНИЯми не стоит ни **--and|-a**, ни **--or|-o**, молчаливо принимается один из операторов. Какой из них **-a** или **-o** - зависит от значения переменной конфигурации **DEFAULT\_IS\_OR**.

2. Операторы **-a** и **-o** имеют одинаковый приоритет, а содержащая их строка имеет левую ассоциативность. Иными словами, **X -o Y -a Z** не будет выполнено, если не выполнено **Z**.

3. **--not|-!** отрицает только результат следующего элемента ФИГУРНАЯ\_СКОБКА\_ИЛИ\_ПРОВЕРКА.

4. Если ШАБЛОН опущен, по умолчанию используется пустой ШАБЛОН. Например, при стандартных настройках **eix**, будучи вызван без аргументов, выведет все пакеты, поскольку каждое имя содержит пустую строку. С другой стороны, **eix -e**, как правило, не должен выводить ничего, ведь не существует пакета с именем, в точности совпадающим с пустой строкой.

5. Заметьте, синтаксис подразумевает, что ШАБЛОН всегда завершает выражение.

КРИТЕРИИ\_ПРОВЕРКИ после ШАБЛОНа всегда начинают новое выражение (т.е. неявно подставляется **--and** или **--or**, в зависимости от значения переменной **DEFAULT\_IS\_OR**). Так, команда **eix -e foo** не эквивалентна **eix foo -e**. Вторая запись означает то же, что и **eix foo --and -e** или **eix foo --or -e**, в зависимости от значения **DEFAULT\_IS\_OR**.

6. Имейте в виду, что КРИТЕРИИ\_ПРОВЕРКИ могут включать несколько опций. Все они применяются одновременно, в том смысле, что они соединены логическим **и** (каково бы ни было значение переменной **DEFAULT\_IS\_OR**). Здесь присутствует некоторая двусмысленность, поскольку ШАБЛОН можно опустить. Во избежание этой двусмысленности последовательные КРИТЕРИИ\_ПРОВЕРКИ всегда рассматриваются как часть одной ПРОВЕРКИ\_ПО\_КРИТЕРИЯМ. Например, в команде **eix -I -O -e foo** все опции рассматриваются как часть одного ВЫРАЖЕНИЯ (а не четырех, как это было бы в случае записи **eix -I " -O " -e " foo**). С другой стороны, в команде **eix -I --not -e** оператор **--not** заставит следующий КРИТЕРИЙ\_ПРОВЕРКИ **-e** воспринимать как относящийся к новому ВЫРАЖЕНИЮ. Опции, отличные от КРИТЕРИЕВ\_ПРОВЕРКИ и логических операторов (таких как **!-, -(, -, -a, -o**), здесь игнорируются. Например, **eix -I -c -e** генерирует только одно ВЫРАЖЕНИЕ, поскольку **-c** не является ни КРИТЕРИЕМ\_ПРОВЕРКИ, ни логическим оператором, а следовательно, не влияет на интерпретацию ВЫРАЖЕНИЯ.

7. КРИТЕРИИ\_ПРОВЕРКИ могут определять **алгоритм соответствия и выбор операции**. Они относятся **только** к текущей ПРОВЕРКЕ\_ПО\_КРИТЕРИЯМ - в частности, они включены только для следующего ШАБЛОНА.

Знайте, что выражения можно использовать как в описанном синтаксисе, так и для неявного отбора пакетов по другим критериям, пусть это и займет дополнительное время. Для этого следует определить **FORMATSTRING** (см. ниже) с использованием условных выражений так, чтобы eix для нежелательных пакетов выводила пустую строку.

Вот допустимые КРИТЕРИИ\_ПРОВЕРКИ:

**-I, --installed**

Искать только среди установленных пакетов. Не используйте эту опцию вместо **eix-installed -a**, **qlist -ICv** или **equery** – они дают разные результаты. В данном случае в выводе не окажется пакетов, которые установлены, но были исключены из дерева портежей или оверлеев; впрочем, их лучше вовсе не иметь в системе (разумно будет помещать их в оверлеи на случай, если вдруг потребуется переустановка). Для обнаружения таких пакетов вы можете использовать команду **eix -te** (или **eix -tI** для включения в вывод и установленных пакетов из портежей), но имейте в виду, что обычные правила **FORMAT** не распространяются на **eix -t**. Таким образом, не следует включать эту опцию в сценариях, если вы не вполне уверены в своих действиях.

Если вы все же решите использовать ее в сценарии вместо **equery**, есть смысл делать это вместе с одной из следующих опций:

**--format --only-names**  
**--format " --pure-packages**  
**-i, --multi-installed**

Искать только среди пакетов, имеющих как минимум две разных установленных версии. Как правило, это означает, что версии были помещены в разные слоты (во время установки).

**-d, --dup-packages**

Искать только среди дублирующих друг друга пакетов: например, `sys-foo/bar` может быть доступен как в официальном дереве портежей, так и в локальном оверлее. Если включена переменная **DUP\_PACKAGES\_ONLY\_OVERLAYS** (см. ниже), искомые пакеты должны находиться в двух разных оверлеях.

**-D, --dup-versions**

Искать только среди пакетов с дублирующими друг друга версиями: например, `sys-foo/bar-0.2.1` может быть доступен как в официальном дереве портежей, так и в локальном оверлее. Если включена переменная **DUP\_VERSIONS\_ONLY\_OVERLAYS** (см. ниже), искомые пакеты должны оба находиться в оверлеях.

**-1, --slotted**

Искать только среди пакетов с непустым слотом, т.е. с непустым и отличным от "0" значением **SLOT**.

**-2, --slots**

Искать только среди пакетов с по крайней мере двумя различными слотами. В отличие от опции **-1**, здесь, если доступен только один слот, например, "4.3", пакет не будет отображен.

**-u, --upgrade, --upgrade+, --upgrade-**

Искать только среди пакетов, имеющих как минимум одну установленную версию пакета в слоте, которая не является лучшей версией в этом слоте. Как правило, это означает, что вам следует либо обновить пакет, либо откатиться до более ранней версии.

Впрочем, проверка будет произведена с учетом значения переменной **UPGRADE\_TO\_HIGHEST\_SLOT** (см. ниже).

Если вы использовали опции **--upgrade+** или **--upgrade-**, операция осуществляется так, как если бы **LOCAL\_PORTAGE\_CONFIG** имела значение **true** или **false**. Иначе утилита будет исходить из значения переменной **UPGRADE\_LOCAL\_MODE**.

Если вы хотите, чтобы отображались только пакеты, рекомендуемые для отката, вам следует обратиться к функциям **FORMATSTRING** (описание см. ниже).

**--stable, --testing, --non-masked, --system, --system-plain+**

Искать только среди пакетов, хотя бы одна версия которых, соответственно, является стабильной (и не замаскированной), тестируемой или стабильной (и не замаскированной), не замаскированной, системной или виртуальным пакетом, только системной. Если в одном запросе вы совмещаете несколько опций из этого ряда, будет отображена только та версия, которая удовлетворяет им всем.

#### --stable+, --testing+, --non-masked+, --system+, --system-plain+

Аналогично описанному выше, но временно принимается значение переменной **LOCAL\_PORTAGE\_CONFIG=true**.

#### --stable-, --testing-, --non-masked-, --system-, --system-plain-

Аналогично описанному выше, но временно принимается значение переменной **LOCAL\_PORTAGE\_CONFIG=false**.

#### --installed-unstable, --installed-testing, --installed-masked

Искать только среди пакетов, у которых установлена хотя бы одна версия, соответственно, нестабильная, тестируемая или замаскированная (временно принимается значение переменной **LOCAL\_PORTAGE\_CONFIG=false**). Если в одном запросе вы совмещаете несколько опций из этого ряда, будет выведена только та версия, которая удовлетворяет им всем.

#### --world, --world-plain

Искать только среди пакетов сета @world (а при **--world** - и среди пакетов-виртуалов для @world). Сопоставимо с "emerge @world", так как включает в поиск не только пакеты, перечисленные в файле world, но и пакеты из world-сетов и сета @system. Если вам это не нужно, выберите другую опцию для более узкой выборки.

#### --world-file, --world-plain

Искать только среди пакетов, фигурирующих в файле world или сети @system. При использовании опции **--world-file** будут отображаться и соответствующие им виртуальные пакеты.

#### --world-set, --world-set-plain

Искать только среди пакетов из файла world\_sets или из сета @system. При использовании опции **--world-set** будут отображаться и соответствующие им виртуальные пакеты.

#### --selected, --selected-plain

Искать только среди пакетов из сета @selected. Сопоставимо с "emerge @selected", так как включает в поиск не только пакеты, перечисленные в файле world, но и пакеты из world-сетов и сета @system. (Если пакеты из сета @system у вас включены в файл world\_sets, то вывод команды, разумеется, будет идентичен выводу при опции **--world**). При использовании опции **--selected** в вывод будут включены соответствующие пакеты-виртуалы. Если вам это не нужно, выберите другую опцию для более узкой выборки.

#### --selected-file, --selected-file-plain

Искать только среди пакетов из файла world. При использовании опции **--selected-file** в вывод будут включены также соответствующие виртуальные пакеты.

#### --selected-set, --selected-set-plain

Искать только среди пакетов из файла world\_set. При использовании опции **--selected-set** в вывод будут включены также соответствующие виртуальные пакеты.

#### --binary

Искать только среди пакетов с бинарным файлом (\*.tbz2-архивом) в PKGDIR. Версия бинарного файла должна совпадать с доступной либо установленной версией пакета. (Заметьте, однако, что если доступной версии нет, пакет также не будет обнаружен.) Проверяется только существование соответствующего \*.tbz2-архива. Может ли portage использовать его - это уже другой вопрос, и ответ на него зависит также от состояния метаданных в \*.tbz2 (например, от настроек USE), но eix за это не отвечает.

#### -O, --overlay

Искать только среди пакетов с как минимум одной версией в оверлее.

#### --in-overlay оверлей

Искать только среди пакетов с как минимум одной версией в оверлее, удовлетворяющем шаблону *оверлей* в аргументе.

Допускаются множественные аргументы: повторите опцию нужное количество раз, указав таким образом все *оверлеи*, которые хотите включить в поиск.

*оверлей* может представлять собой либо шаблон, либо число. Обратите внимание, что, используя предлагаемое по умолчанию значение переменной **OVERLAYS\_LIST=all-used-renumbered**, вы не увидите корректной нумерации оверлеев; чтобы узнать правильный номер интересующего вас оверлея, необходимо вызвать:

#### **OVERLAYS\_LIST=all eix --not**

...а в сценариях - лучше так:

#### **OVERLAYS\_LIST=all PRINT\_COUNT\_ALWAYS=never eix -!**

Специальные значения **0** и **\$PORTDIR** соответствуют основному дереву портежей (оно условно принимается за нулевой оверлей).

Если аргумент *оверлей* пуст (или опущен, если последняя опция **--in-overlay**), поиск будет произведен по всем оверлеям, кроме основного дерева портежей (т.е. **--in-overlay ''** работает идентично **-O**).

#### --only-in-overlay оверлей

Искать среди пакетов с версиями только в оверлее, соответствующем шаблону *оверлей*.

Допускаются множественные аргументы: повторите опцию нужное количество раз, указав таким образом все *оверлеи*, которые хотите включить в поиск.

*оверлей* может представлять собой либо шаблон, либо число, аналогично **--in-overlay**. **--only-in-overlay ''** выведет все соответствующие аргументу пакеты не из официального дерева портежей, доступные только в оверлеях.

#### **-J, --installed-overlay**

Искать только среди пакетов, установленных из какого-либо оверлея. Чтобы обеспечить надежность результатов, укажите значением переменной **CHECK\_INSTALLED\_OVERLAYS** true (это значение не выставлено по умолчанию, потому что значительно замедляет работу утилиты). Подробнее см. в описании переменной **CHECK\_INSTALLED\_OVERLAYS**.

#### --installed-from-overlay оверлей

Эта опция аналогична **--in-overlay** с той разницей, что поиск будет произведен только среди пакетов, хотя бы одна версия которых установлена из оверлея. Например, по запросу **--installed-from-overlay 0** будут выведены только те пакеты, хотя бы одна версия которых была установлена из дерева портежей. Чтобы обеспечить надежный результат при использовании **-J**, выставьте значением переменной **CHECK\_INSTALLED\_OVERLAYS** true.

#### --installed-in-some-overlay

Искать только среди пакетов с по крайней мере одной установленной версией, доступной также в оверлеях.

#### --installed-in-overlay оверлей

Опция аналогична **--in-overlay** с той разницей, что поиск ведется только среди пакетов, хотя бы одна установленная версия которых существует в оверлее. Так, если задать **--installed-in-overlay 0**, будут выведены только те пакеты, у которых установлена хотя бы одна версия, доступная также в основном дереве портежей.

#### --restrict-fetch

Искать только среди пакетов, у которых хотя бы для одной версии RESTRICT=fetch. При использовании в связке с другими критериями проверки PROPERTIES/RESTRICT выводится только та версия, которая удовлетворяет им всем.

#### **--restrict-mirror**

Искать только среди пакетов, у которых хотя бы для одной версии RESTRICT=mirror. При использовании в связке с другими критериями проверки PROPERTIES/RESTRICT выводится только та версия, которая удовлетворяет им всем.

#### **--restrict-primaryuri**

Искать только среди пакетов, у которых хотя бы для одной версии RESTRICT=primaryuri. При использовании в связке с другими критериями проверки PROPERTIES/RESTRICT выводится только та версия, которая удовлетворяет им всем.

#### **--restrict-binchecks**

Искать только среди пакетов, у которых хотя бы для одной версии RESTRICT=binchecks. При использовании в связке с другими критериями проверки PROPERTIES/RESTRICT выводится только та версия, которая удовлетворяет им всем.

#### **--restrict-strip**

Искать только среди пакетов, у которых хотя бы для одной версии RESTRICT=strip. При использовании в связке с другими критериями проверки PROPERTIES/RESTRICT выводится только та версия, которая удовлетворяет им всем.

#### **--restrict-test**

Искать только среди пакетов, у которых хотя бы для одной версии RESTRICT=test. При использовании в связке с другими критериями проверки PROPERTIES/RESTRICT выводится только та версия, которая удовлетворяет им всем.

#### **--restrict-userpriv**

Искать только среди пакетов, у которых хотя бы для одной версии RESTRICT=userpriv. При использовании в связке с другими критериями проверки PROPERTIES/RESTRICT выводится только та версия, которая удовлетворяет им всем.

#### **--restrict-installsources**

Искать только среди пакетов, у которых хотя бы для одной версии RESTRICT=installsources. При использовании в связке с другими критериями проверки PROPERTIES/RESTRICT выводится только та версия, которая удовлетворяет им всем.

#### **--restrict-bindist**

Искать только среди пакетов, у которых хотя бы для одной версии RESTRICT=bindist. При использовании в связке с другими критериями проверки PROPERTIES/RESTRICT выводится только та версия, которая удовлетворяет им всем.

#### **--restrict-parallel**

Искать только среди пакетов, у которых хотя бы для одной версии RESTRICT=parallel. При использовании в связке с другими критериями проверки PROPERTIES/RESTRICT выводится только та версия, которая удовлетворяет им всем.

#### **--properties-interactive**

Искать только среди пакетов, у которых хотя бы для одной версии PROPERTIES=interactive. При использовании в связке с другими критериями проверки PROPERTIES/RESTRICT выводится только та версия, которая удовлетворяет им всем.

#### **--properties-live**

Искать только среди пакетов, у которых хотя бы для одной версии PROPERTIES=live. При использовании в связке с другими критериями проверки PROPERTIES/RESTRICT выводится только та версия, которая удовлетворяет им всем.

#### --properties-virtual

Искать только среди пакетов, у которых хотя бы для одной версии PROPERTIES=virtual. При использовании в связке с другими критериями проверки PROPERTIES/RESTRICT выводится только та версия, которая удовлетворяет им всем.

#### --properties-set

Искать только среди пакетов, у которых хотя бы для одной версии PROPERTIES=set. При использовании в связке с другими критериями проверки PROPERTIES/RESTRICT выводится только та версия, которая удовлетворяет им всем.

#### -T, --test-obsolete

Искать только среди устаревших пакетов.

Пакеты считаются устаревшими, если им соответствуют дублирующие друг друга записи в /etc/portage/package.\* (при условии, что переменная **TEST\_FOR\_REDUNDANCY** имеет значение true) или если не все установленные версии существуют (при условии, что переменная **TEST\_FOR\_NONEXISTENT** имеет значение true).

Определение дублирования содержится в переменных **REDUNDANT\_IF**, описанных ниже, а определение несуществования - в переменных **NONEXISTENT\_IF**. Обратите внимание, что проверка версий пакетов из устаревших оверлеев надежна только в том случае, если переменная **CHECK\_INSTALLED\_OVERLAYS** у вас имеет значение true (это не значение по умолчанию, поскольку при его использовании проверка значительно замедляется). Подробнее см. в описании переменной **CHECK\_INSTALLED\_OVERLAYS**.

Учтите, что данная опция обеспечивает проверку только тех пакетов, которых находятся в базе данных - в частности, при ее использовании вы не обнаружите в выводе записей, соответствующих переименованным или удаленным пакетам (из дерева портежей). Если вы хотите увидеть именно их, используйте **-t**.

Итак, для поиска устаревших записей другого рода целесообразно применять данную опцию в связке с **-t**.

Если по какой-либо причине вы хотите исключить из проверки некоторые пакеты, вы можете внести соответствующие записи в файл (или каталог) /etc/portage/package.nowarn. Он будет описан ниже; там же вы найдете информацию о том, как указывать альтернативные/дополнительные файлы).

#### -|, --pipe

(Помните, что интерпретатор команд не обработает символ | без кавычек.)

Искать только среди пакетов из стандартного ввода. Как правило, эту опцию удобно использовать в конвейере, например, перенаправляя вывод emerge -rv (аналогично genlop -r). Будут обработаны все данные, формат которых содержит следующие слова (через пробел или знак новой строки):

категория/пакет-версия или

категория/пакет

Кроме того, все пакеты/версии, обработанные таким образом, в выводе будут маркированы. Подробнее о маркировании см. в описании строк формата **marked** и **markedversions:\***.

Даже если опция **--pipe** встречается несколько раз, стандартный ввод, разумеется, будет считан лишь единожды, но каждое вхождение опции обрабатывается отдельно (т.е. если первая **--pipe** выдает соответствие, то выдаст соответствие и остальные).

Если вы хотите использовать стандартный ввод только для маркирования, но не для выбора, можно использовать выражение вида

**eix something -a "(-" --pipe -o "-)"**

## Выбор поля соответствия

Следующие опции определяют поля критериев, на соответствие которым будет проверяться шаблон.

Одно выражение может включать несколько полей (выражение удовлетворяет запросу, если шаблон удовлетворяет хотя бы одному из указанных полей). Если вы не укажете некоторые из приведенных опций, по умолчанию будет выбрано поле соответствия **--name**, но если у вас особый шаблон - например, вида "категория/пакет" или "@сет", вместо этого по умолчанию будет использовано поле соответствия **--category-name**, **--set**, **--description**, **--homepage**, **--virtual** или **--license**. Параметры эвристической процедуры определяются переменной конфигурации **DEFAULT\_MATCH\_FIELD** (подробнее о ней см. ниже).

### **-s, --name**

В аргументе, например, eix.

### **-S, --description**

В аргументе, например, Small utility for searching.

### **-C, --category**

В аргументе, например, app-portage.

### **-A, --category-name**

В аргументе, например, app-portage/eix.

### **-H, --homepage**

В аргументе, например, <http://eix.berlios.de/>.

### **-L, --license**

В аргументе, например, GPL-2.

### **-P, --provide**

В аргументе, например, virtual/blackbox.

### **--set**

В аргументе - имя локального сета пакетов версии, содержащейся в базе данных (т.е. соответствующей файлу в каталоге **/etc/portage/sets**, **/etc/portage/sets.eix**, или другом каталоге из значений переменной **EIX\_LOCAL\_SETS\_ADD**; см. комментарии к **EIX\_LOCAL\_SETS**). Сеты **system** и **world** здесь преднамеренно не учитываются; для их сканирования следует использовать опции **--system[+-]**, **--world**, **--world-all** и **--world-sets**.

### **--slot**

В аргументе - имя слота версии в базе данных, например, kde-4.

### **--installed-slot**

В аргументе - имя слота установленной версии. Помните, что без опции **--care** (или выставленного значения переменной **CAREMODE=true**) имя слота может быть только предположено.

### **-U, --use**

В аргументе - USE-флаг, определенный IUSE в некоторой версии некоторыми билдами пакета. Чаще всего эта опция используется вместе с **-e**.

### **--installed-with-use**

В аргументе - USE-флаг, который был включен при установке пакета. Разумеется, поиск в этом случае ведется только среди установленных пакетов. Заметьте, что это относится и к опции **-I** - в обоих случаях в обработку будут включены только пакеты, в настоящий момент присутствующие в базе данных.

### **--installed-without-use**

В аргументе - USE-флаг, который был отключен при установке пакета. Разумеется, поиск в этом случае ведется только среди установленных пакетов. Заметьте, что это относится и к опции **-I** - в обоих случаях в обработку будут включены только пакеты, в настоящий момент присутствующие в базе данных.

## Алгоритм соответствия

Ниже следующие опции определяют алгоритм, по которому поля соответствия будут соотнесены с шаблоном. Для одного соответствия можно выбрать только один алгоритм. Если вы опустите какие-либо из этих опций, значение по умолчанию будет определено эвристически, в зависимости от вида шаблона, по которому ведется поиск. В большинстве случаев по умолчанию будет использована опция **--regex**, за исключением случаев, когда вид шаблона говорит о том, что это glob-шаблон или подстрока (тогда по умолчанию используется соответствующий алгоритм), или когда шаблон имеет особый вид, например, или *категория/пакет* или *@сет* - в этом случае большинство пользователей ожидают поиска по всей строке или, соответственно, по началу строки. Параметры эвристической процедуры определяются переменной конфигурации **DEFAULT\_MATCH\_ALGORITHM** (подробнее о ней см. ниже).

### **-e, --exact**

В аргументе - точная (полная) строка шаблона. Например, команда `eix -e gcc` выведет только пакеты gcc.

### **-b, --begin**

Шаблон находится в начале строки. Так, команда `eix -b gcc` выведет не только пакет gcc, но и, например, gcc-config.

### **--end**

Шаблон находится в конце строки.

### **-z, --substring**

Шаблон находится в пределах строки.

### **-f [N], --fuzzy [N]**

Будет произведен приблизительный поиск с максимальным расстоянием Левенштейна *N* (по умолчанию) для всей строки. Имейте в виду, что использование этой опции замедляет поиск.

### **-p, --pattern**

В аргументе - подстановочный шаблон (для всей строки). Подробности см. в справке по **fnmatch(3)** и/или **glob(7)**. Убедитесь, что шаблоны заключены в одинарные кавычки (чтобы оболочка не перехватывала подстановочные знаки).

### **-r, --regex**

В аргументе - регулярное выражение. Оно ищется только как подстрока (если только не используются символы ^, \$); пустой шаблон вызывает вывод всех пакетов. Более подробную информацию об этом вы найдете в man-руководстве **regex(7)**. Как и в предыдущем случае, убедитесь, что шаблон заключен в одинарные кавычки.

## Определение формата вывода (см. ниже FORMATSTRING)

### **--format ФОРМАТ**

Назначить **ФОРМАТ** стандартным форматом вывода результатов поиска.

### **--format-compact ФОРМАТ**

Назначить **ФОРМАТ** компактным форматом вывода результатов поиска.

### **--format-verbose ФОРМАТ**

Назначить **ФОРМАТ** подробным форматом вывода результатов поиска.

## Особые опции для eix-update

### **-o файл\вывода, --output файл\_вывода\_**

При использовании этой опции **eix-update** запишет базу данных eix не в **/var/cache/eix**, а в **файл\вывода\_**, не проверяя и не меняя права доступа к этому файлу.

#### **-a оверлей, --add-overlay** оверлей

Эта опция аналогична добавлению **оверлея** в значения переменной **PORTDIR\_OVERLAY** в файле **/etc/make.conf** или изменению **ADD\_OVERLAY**, но имеет то преимущество, что вам нет нужды редактировать переменные, а кроме того, вы можете использовать пробелы в аргументе. Оверлеи, добавленные с помощью данной опции, идут следом за оверлеями, добавленными через **KEEP\_VIRTUALS**. Если **оверлей** уже фигурирует в списке оверлеев, эта опция отработает вхолостую. Опция может быть использована несколько раз, для добавления нескольких оверлеев.

#### **-x оверлей, --exclude-overlay** оверлей

Эта опция аналогична добавлению **оверлея** в значения переменной **EXCLUDE\_OVERLAY**, но имеет то преимущество, что вам нет нужды редактировать переменную, а кроме того, вы можете использовать пробелы в аргументе. **оверлей** здесь рассматривается как маска. Все удовлетворяющие шаблону оверлеи (даже те, которые были добавлены с помощью описанных ниже опций **--add-overlay**) исключаются из списка оверлеев. Каталог **PORTDIR** рассматривается как обычный оверлей, который также может быть исключен (в таком случае в качестве **PORTDIR** будет сохранен первый **оверлей** из перечисленных). Опция может быть использована несколько раз, для исключения нескольких оверлеев.

#### **-m оверлей метод, --override-method** оверлей метод

Изменить текущий метод кэширования **оверлея** (каталог **PORTDIR** считается допустимым оверлеем) на указанный аргументом **метод**. **оверлей** рассматривается как маска, т.е. может содержать подстановочные знаки. Если **оверлей** не сопоставлен ни одной записи в списке оверлеев, эта опция отработает вхолостую. Данная опция аналогична добавлению записи "**оверлей метод**" в конец переменной **OVERRIDE\_CACHE\_METHOD**. Опция может быть использована несколько раз, для переопределения метода кэширования для нескольких оверлеев. Приоритет имеет последнее значение. В частности, значение, переопределенное с помощью данной опции, будет иметь приоритет над значением **OVERRIDE\_CACHE\_METHOD**.

#### **-r путь-к-оверлею метка-оверлея, --repo-name** путь-к-оверлею метка-оверлея

Оверлей, расположенный по адресу **путь-к-оверлею**, получит **метку-оверлея**, независимо от других настроек. Это поведение можно переопределить через **REPO\_NAMES**. В отличие от значения **REPO\_NAMES**, **путь-к-оверлею** представляет собой не шаблон, а точный путь.

## **ВЫВОД**

### **Слоты**

В отличие от обычного вывода версий в **emerge**, **eix** может отображать также имена слотов, если они непусты и отличны от нуля. Это поведение определяется содержимым **FORMATSTRING**.

Если слоты отображаются, то имя слота либо отделяется от номера версии двоеточием, либо заключается в скобки. Вы можете выбрать предпочтительный формат разделителя, редактируя переменную **COLON\_SLOTS**.

Если переменная **PROPERTIES** или **RESTRICT** определена в ебилде, это по умолчанию показывается в строке версии; более тонкая настройка доступна путем редактирования переменных конфигурации.

Вот несколько примеров:

#### **4.1.1:4.1** или **4.1.1(4.1)**

Версия 4.1.1 будет установлена в слот 4.1.

#### **3.14p:GNAT-3.14p** или **3.14p(GNAT-3.14p)**

Версия 3.14p будет установлена в слот GNAT-3.14p.

#### **2.0.0\_rc1-r6**

Версия 2.0.0\_rc1-r6, значение SLOT либо пусто, либо "0".

## **1.0+i+l+v+s!f!m!p!b!s!t!u!i!d!P{tbz2}**

Версия 1.0, для которой PROPERTIES="interactive live virtual set", а также RESTRICT="fetch mirror primaryuri binchecks strip test userpriv installsources bindist parallel" Кроме того, в **PKGDIR**, существует \*.tbz2-архив (бинарный пакет) для этой версии.

## **5.0-r3(5.0R3)!f**

Версия 5.0-r3 будет установлена в слот 5.0R3, с ограничением по загрузке.

## **Маскировка**

Достаточно поработать с gentoo неделю, чтобы легко опознавать формат маскировки в строках версий. Тем не менее мы напомним об этом на нескольких примерах. Разумеется, изложенное ниже относится только к стандартным настройкам; чтобы изменить параметры, используйте параметры конфигурации.

### **[P]2.95.3-r8**

Если в файлах пакетов вашего профиля была обнаружена маска пакета, но данная версия не совпадает с ней, говорят, что версия "замаскирована профилем".

### **[M]4.0.0\_alpha20050213**

Версия совпадает с маской из /etc/portage/package.mask, \$PORTDIR/profiles/package.mask или package.mask в вашем профиле. Portage называет это "маскировкой package.mask".

### **[m]4.1.4**

Версия совпадает с локальной маской (из /etc/portage/package.mask), но не замаскирована ни профилем, ни \$PORTDIR/profiles/package.mask.

### **{P}2.95.3-r8**

Первоначально версия была замаскирована профилем, но эта маскировка была локально изменена в /etc/portage/profile/packages.

### **{M}4.0.0\_alpha20050213**

Первоначально версия была замаскирована \$PORTDIR/profiles/package.mask, но эта маскировка была локально изменена в /etc/portage/package.unmask.

### **\*3.3.3**

Версия "замаскирована отсутствующим ключевым словом", но стабильна на других архитектурах.

### **~\*3.3.3**

Версия замаскирована отсутствующим ключевым словом и нестабильна на других архитектурах.

### **\*\*3.3.3**

Версия замаскирована отсутствующим ключевым словом для всех архитектур.

### **(\*\*)3.4.3-r2**

Первоначально версия не имела ключевого слова, но этот параметр был локально изменен (в /etc/portage/package.keywords или путем редактирования переменной ACCEPT\_KEYWORDS).

### **-\*3.4.3-r2**

Версия "замаскирована ключевым словом -\*" для всех архитектур (в скором времени это обозначение перестанет поддерживаться).

### **-0.8.14**

Версия замаскирована -ARCH.

### **~3.3.5.20050130**

Версия замаскирована ключевым словом ~keyword.

### **(~)3.3.5.20050130**

Первоначально версия была замаскирована ключевым словом `~keyword`, но эта маскировка была локально изменена (в `/etc/portage/package.keywords` или путем редактирования переменной `ACCEPT_KEYWORDS`).

### [M]~1.0.9626

Версия замаскирована одновременно `package.mask` и ключевым словом `~keyword`.

### [m](~)4.1.4-r1

Первоначально версия была замаскирована только ключевым словом `~keyword`, но эта маскировка была локально изменена (в `/etc/portage/package.keywords` или путем редактирования `ACCEPT_KEYWORDS`). Тем не менее версия замаскирована локально (в `/etc/portage/package.mask`).

## 3.3.1

Наконец, эта запись обозначает стабильную версию; она стабильна и без локальных настроек.

### eix-diff

Вывод утилиты **eix-diff** полностью определяется переменными конфигурации (`DIFF_FORMAT_NEW`, `DIFF_FORMAT_DELETE`, `DIFF_FORMAT_CHANGED` и множество других переменных, к которым - по крайней мере, при стандартных настройках - обращаются перечисленные посредством отложенной замены. См. ниже.) Таким образом, ниже на примерах мы показываем только стандартное поведение текущей версии eix. Хотя этот стандартный формат уже довольно давно не меняется, в новых версиях eix мы не можем гарантировать стабильности настроек по умолчанию.

#### [N] >> **foo/bar (~1.0): description of foo/bar**

В дереве портежей появился свежий пакет **foo/bar**.

#### [\*N] >> **foo/bar (1.0): description of foo/bar**

В дереве портежей появился свежий пакет **foo/bar**. Кроме того, он имеет версию (1.0), которая может быть установлена без размаскировки или изменения ключевых слов.

#### << **foo/bar ({M}1.0): description of foo/bar**

Пакет **foo/bar** был удален из дерева портежей; предыдущая версия, **1.0**, ранее была замаскирована, но сейчас уже не замаскирована (вероятно, потому, что разработчик при удалении пакета удалил файл `package.mask`).

#### [\*>] == **foo/bar (1.0): description of foo/bar**

Статус пакета **foo/bar** в дереве портежей изменился (для ваших настроек): для него появилась версия (1.0), которая может быть установлена без размаскировки и изменения ключевых слов, в то время как ранее **foo/bar** такой версии не имел. Кроме того, символ **>** означает, что один слот получил более новую версию. В данном случае появление и **>**, и **\*** вызвано одним и тем же изменением.

#### [><] == **foo/bar (1.1(1) 2.0(2) -> 1.0(1) 2.1(2)): description**

Статус пакета **foo/bar** в дереве портежей изменился (для ваших настроек): символы в левой части строки означают, что один слот получил более новую версию пакета, которая может быть установлена без размаскировки и изменения ключевых слов, а из другого слота эта новая версия была удалена. Если вы посмотрите на версии, то увидите, что слот **2** получил новую версию (предыдущая стабильная версия в этом слоте была **2.0**, теперь **2.1**), а версия, которая до этого была последней, **1.1**, была удалена из слота **1** или замаскирована (текущая стабильная версия в этом слоте - **1.0**).

#### [UD] == **foo/bar (1.1(1)@01.01.2009; 1.1(1) -> 2.0(2)): description**

Статус пакета **foo/bar** в дереве портежей изменился (для ваших настроек): символы в левой части строки означают, что единственный установленный слот может быть обновлен (без изменения масок/ключевых слов), а другой слот, в который производилась установка, был удален/замаскирован. Версии, указанные в правой части строки, показывают, что установленная версия **1.1** в слоте **1** была удалена или замаскирована, а другой установленной версии в **1** нет. Однако в слоте **2** появилась новая стабильная версия (слот **2** ранее не существовал или не имел стабильной версии).

Поскольку еще ни одна версия не была установлена в слот **2**, ей в этой ситуации не может определить, уместно ли обозначение "**U**". Ведь ей не отслеживает зависимости и поэтому не знает, будет ли использован новый слот, например, world-файлом, или же существует только некоторая зависимость со старым слотом. Поэтому символ "**U**" в этом случае будет отображен только если выставлено значение **UPGRADE\_TO\_HIGHEST\_SLOT=true** или если пакет фигурирует в **/etc/portage/package.slot\_upgrade\_allow**.

Вывод вида == **foo/bar** ... был бы, на самом деле, более логичным, поскольку дополнительно один слот приобрел новую стабильную версию, а из другого слота была удалена версия, до бывшая до того времени новейшей стабильной. Однако, поскольку "**U**" или, соответственно, "**D**" это и так подразумевают, разработчики приняли решение по умолчанию никогда не отображать символы < и >, если имеется обозначение **U** или **D**. Разумеется, вы вправе изменить это поведение, создав собственную строку **DIFF\_FORMAT\_HEADER\_CHANGED**.

## FORMATSTRING

Строка формата может содержать условные блоки, свойства пакетов, настройки цветного вывода и стандартные строки. Если для какого-либо пакета строка формата занимает часть пустой строки, отображается и завершающий переход на новую строку. Таким образом, вы можете заключить всю строку формата в условный блок, чтобы выводились только те пакеты, которые соответствуют условию. Ниже будет приведен пример такого сценария-обертки.

### Условные блоки

Принцип действия условий прост: расширяется какое-либо свойство, и результирующая строка сопоставляется с другой строкой. Если они совпадают, условие признается истинным и выполняются команды в блоке. С условиями можно использовать отрицание, чтобы, когда условие истинно, выполнялись команды из части "else", а когда оно ложно - команды из части "if". Часть "else" может не выполняться вовсе.

{[!]СВОЙСТВО[=RHS]}TCODE{}

Если строка, полученная в результате развертывания **СВОЙСТВА**, совпадает с **RHS**, выполнить **TCODE**. Символ ! обозначает отрицание действия. **RHS** представляет собой либо свойство (если заключен в <>), либо переменную (если имеет префикс \$), либо строку (если заключен в кавычки или не сопровождается никакими служебными символами).

{[!]СВОЙСТВО[=RHS]}TCODE{else}FCODE{}

Если строка, полученная в результате развертывания **СВОЙСТВА**, совпадает с **СТРОКОЙ**, выполнить **TCODE**. Иначе выполнить **FCODE**.

**СВОЙСТВО** может быть либо одним из свойства пакетов, описанных ниже, либо обращением к переменной. Обращение к переменной имеет вид **\$ПЕРЕМЕННАЯ**. Нет необходимости инициализировать эту **ПЕРЕМЕННУЮ** : по умолчанию она содержит пустую строку.

Для изменения **ПЕРЕМЕННОЙ** исполняемой среды используйте следующий синтаксис:

{[!]\*ПЕРЕМЕННАЯ[=RHS]}

Значением переменной исполняемой среды **ПЕРЕМЕННАЯ** станет **СТРОКА**. С ! результат будет либо пустым, либо равным 1, в зависимости от того, пуста или непуста **СТРОКА**. Если опущена завершающая часть (включая символ =), блок получает особое значение: {\*!<ПЕРЕМЕННАЯ>} устанавливает значением **ПЕРЕМЕННОЙ** единицу, {\*!<ПЕРЕМЕННАЯ>} устанавливает значением **ПЕРЕМЕННОЙ** пустую строку.

## Свойства пакетов

Имена, которые соответствуют отдельным свойствам обрабатываемого пакета. Если вы используете имя для вывода свойства, оно **должно быть заключено в "уголки"** (т.е. иметь вид "<имя>").

**name, category, homepage, licenses**

Соответственно название, категория, веб-страница проекта текущего пакета и лицензии, под которыми он распространяется.

#### **availableversions:ПЕРЕМЕННАЯ, availableversions:ПЕРЕМЕННАЯ:СЛОТЫ\ПЕРЕМЕННЫХ\_**

Для каждой версии выводится содержимое переменной конфигурации/окружения с именем **ПЕРЕМЕННАЯ**; оно интерпретируется как строка формата. При использовании второго типа синтаксиса и если хотя бы один слот пакета непуст, вместо **ПЕРЕМЕННОЙ** работают **СЛОТЫ\ПЕРЕМЕННЫХ\_**, а версии сортируются по слотам.

На всякий случай уточним: требуемый формат нельзя указывать непосредственно после двоеточия; вместо этого необходимо сохранить его в новой переменной. **ПЕРЕМЕННАЯ** и **СЛОТЫ\ПЕРЕМЕННЫХ\_** - всего лишь их имена.

Полезными примерами **ПЕРЕМЕННОЙ** являются **NAMEVERSION**, **EQNAMEVERSION**, **EQNAMEVERSION**, **ANAMESLOT**, **ANAMEASLOT**, **NAMESLOT**, **NAMEASLOT**, **DATESORT**. **ANAMESLOT** и **ANAMEASLOT** предназначены для использования во втором типе синтаксиса, т.е. в конструкциях **availableversions:ANAMESLOT:ANAMESLOT** или **availableversions:ANAMEASLOT:ANAMEASLOT** (Легко запомнить: **ASLOT** отображает слот всегда (**always**)). **NAMESLOT**, **NAMEASLOT** и **DATESORT** имеют смысл только для установленных версий. Подробнее об этих переменных читайте, выполнив **eix --dump**.

#### **markedversions:ПЕРЕМЕННАЯ, markedversions:ПЕРЕМЕННАЯ:СЛОТЫ\ПЕРЕМЕННЫХ\_**

Аналогично **availableversions** с той разницей, что будут выведены только маркированные версии.

#### **bestversion:ПЕРЕМЕННАЯ, bestversion\*:ПЕРЕМЕННАЯ, bestslotversions:ПЕРЕМЕННАЯ, bestslotversions\*:ПЕРЕМЕННАЯ, bestslotupgradeversions:ПЕРЕМЕННАЯ, bestslotupgradeversions\*:ПЕРЕМЕННАЯ**

Аналогично **availableversions** с той разницей, что будет выведена только лучшая/-ие версия/-ии в каждом слоте. При использовании конструкций с символом \* в вывод включаются и нестабильные версии. При использовании конструкций, включающих **upgrade**, обрабатываются будут только те версии, которые должны появиться после обновления.

#### **installedversions:ПЕРЕМЕННАЯ**

Аналогично **availableversions** с той разницей, что будут выведены только установленные версии.

#### **installedmarkedversions:ПЕРЕМЕННАЯ**

Аналогично **installedversions** с той разницей, что будут только маркированные версии.

#### **first, last, slotfirst, slotlast, oneslot**

Только если **ПЕРЕМЕННАЯ** используется в контексте вывода версий. Вы можете использовать эти флаги, чтобы выяснить, первую или последнюю версию пакета вы просматриваете (как правило, это бывает полезно, если вам необходим вывод какого-либо дополнительного текста). Аналогично, если вывод сортируется по слотам, вы можете проверить, первая или последняя версия в слоте отображается; возможно, существует всего один слот. Если условие удовлетворено, все эти свойства пусты, иначе их значение - 1. Чтобы было удобнее повторно использовать код, когда вывод сортируется по слотам, переменная **slotfirst[slotlast]** эквивалентна **first/last**, а значение **oneslot** равно 1.

#### **slot, isslot, overlayver, overlaynum, versionkeywords**

Только если **ПЕРЕМЕННАЯ** используется в контексте вывода версий. Будет выведен текущий слот, оверлей или полные ключевые слова для текущей версии. Переменная **overlayver** пуста, если пакет целиком из одного оверлея; в этом случае, чтобы увидеть оверлей, следует использовать **overlay**.

**overlaynum** же содержит номер оверлея; не поддерживает цветной вывод (если пакет не из оверлея, переменная пуста). Формат вывода **versionkeywords** обусловлен значениями переменных

**FORMAT\_BEFORE\_KEYWORDS**, **FORMAT\_AFTER\_KEYWORDS**,

**PRINT\_EFFECTIVE\_KEYWORDS**, **FORMAT\_BEFORE\_EFFECTIVE\_KEYWORDS**,

**FORMAT\_AFTER\_EFFECTIVE\_KEYWORDS**. **isslot** позволяет выяснить, не пуст ли слот, и в этом случае возвращает 1, в противном случае не возвращает ничего.

## **isbestupgrade, isbestupgrade\*, isbestupgradeslot, isbestupgradeslot\***

Только если *ПЕРЕМЕННАЯ* используется в контексте вывода версий. Если текущая версия является лучшей или, соответственно, лучшей в текущем слоте для обновления, будет возвращена единица. При использовании конструкций, включающих символ \*, в обработку будут включены нестабильные версии.

## **installedversion, markedversion**

Только если *ПЕРЕМЕННАЯ* используется в контексте вывода версий. Возвращает 1, если текущая версия установлена или, соответственно, маркирована, иначе возвращает пустую строку.

## **ishardmasked, washardmasked, isprofilemasked, wasprofilemasked, ismasked, wasmasked, isstable, wasstable, isunstable, wasunstable, isalienstable, wasalienstable, isalienunstable, wasalienunstable, ismissingkeyword, wasmissingkeyword, isminuskeyword, wasminuskeyword, isminusunstable, wasminusunstable, isminusasterisk, wasminusasterisk**

Только если *ПЕРЕМЕННАЯ* используется в контексте вывода версий. Возвращает 1, если текущая версия стабильна в локальной конфигурации или, соответственно, в конфигурации по умолчанию, иначе возвращает пустую строку.

## **isbinary**

Только если *ПЕРЕМЕННАЯ* используется в контексте вывода версий. Если существует соответствующий версии \*.tbz2-архив, возвращает 1, если архив не существует, возвращает пустую строку.

## **restrict, restrictfetch, restrictmirror, restrictprimaryuri, restrictbincheck, restrictstrip, restricttest, restrictuserpriv, restrictinstalledsources, restrictbindist, restrictparallel**

Только если *ПЕРЕМЕННАЯ* используется в контексте вывода версий. Если для версии включен какой-либо из атрибутов **RESTRICT**, возвращает 1, если нет, возвращает пустую строку.

## **properties, propertiesinteractive, propertieslive, propertiesvirtual, propertiesset**

Только если *ПЕРЕМЕННАЯ* используется в контексте вывода версий. Если для версии включен какой-либо из атрибутов **PROPERTIES**, возвращает 1, если нет, возвращает пустую строку.

## **haveuse, use**

Только если *ПЕРЕМЕННАЯ* используется в контексте вывода версий. Отображает информацию сообразно значениям переменной **IUSE** (для доступных версий) или **USE** (для установленных версий). **haveuse** можно использовать для проверки, будет ли вывод непустым (тогда возвращает 1, иначе пустую строку). Для доступных версий **use** выводит переменную **IUSE**. Для установленных версий **use** выводит **USE**-флаги и информацию о том, установлены ли они (если нет, содержимое переменных **FORMAT\_BEFORE\_SET\_USE**, **FORMAT\_AFTER\_SET\_USE**, **FORMAT\_BEFORE\_UNSET\_USE**, **FORMAT\_AFTER\_UNSET\_USE** выводится там, где это предусмотрено).

## **date:*ПЕРЕМЕННАЯ***

Только если *ПЕРЕМЕННАЯ* используется в контексте вывода версий. Отображает дату установки; формат функции даты **strftime()** считывается из *ПЕРЕМЕННОЙ*.

## **version**

Только если *ПЕРЕМЕННАЯ* используется в контексте вывода версий. Выводит простую версию в текстовом формате.

## **installed, best, best\***

Возвращает 1, если хотя бы одна версия пакета установлена или, соответственно, имеется лучшая стабильная/нестабильная версия пакета, иначе возвращает пустую строку.

## **versionlines**

Если утилите был передан флаг **--versionlines**, возвращает 1, иначе возвращает пустую строку.

## **slotsorted**

Если утилите был передан флаг **--versionsort**, возвращает 1, иначе возвращает пустую строку.

## **color**

Если вывод поддерживает цвета/маркеры, возвращает 1, иначе возвращает пустую строку. Например, если вывод перенаправляется в терминал и явно не выставлена обратная опция, переменная будет пуста.

## **setnames, allsetnames**

Имена всех локальных сетов, которым принадлежит пакет, через пробел. При использовании **allsetnames** в обработку будет включен сет system.

## **binary**

Если хотя бы для одной версии (доступной или установленной) имеется соответствующий ей \*.tbz2-архив, возвращает 1, иначе пустую строку. См. примечания к опции **--binary**.

## **overlaykey**

Если все версии находятся в одном оверлее, значение "[overlaykey]" будет выведено с поддержкой цвета.

## **system**

Если пакет существует в системном профиле или является виртуалом пакета из системного профиля, переменная получает значение 1.

## **world**

Если пакет фигурирует в world-файле или является виртуалом пакета из world-файла, переменная получает значение 1.

## **world\_sets**

Если пакет принадлежит world-сетам или является виртуалом пакета из world-сетов, переменная получает значение 1.

## **systempure**

Если пакет существует в системном профиле, переменная получает значение 1.

## **worldpure**

Если пакет фигурирует в world-файле, переменная получает значение 1.

## **world\_setspure**

Если пакет принадлежит world-сетам, переменная получает значение 1.

## **provide**

Строка PROVIDE для пакета.

## **marked**

Если пакет был передан утилите с опцией **--pipe**, переменная получает значение 1. Обычно это имеет смысл только для проверки.

## **havemarkedversion**

Если хотя бы одна доступная версия пакета маркована, возвращает 1, иначе пустую строку. Имейте в виду, что пакет может оказаться маркован, при том что не маркована ни одна из его версий.

## **slots, slotted**

Если имеется как минимум два слота или, соответственно, как минимум один непустой слот, переменная получает значение 1.

## **colliuse, havecolliuse**

Набор IUSE-флагов (т.е. все флаги вместе взятые) для всех доступных версий пакета, через пробел. Переменная **havecolliuse** получает значение 1, если **colliuse** непуста; она работает быстрее, чем **colliuse**.

## **haveversionuse**

Для экономии памяти и/или дискового пространства eix можно собирать без поддержки **IUSE**-флагов для каждой отдельно взятой версии. Если текущий пакет взят из такой базы данных или если eix был скомпилирован без учета этой информации, будет возвращена пустая строка, иначе 1. Даже если проверка дала отрицательный результат, вы можете использовать синтаксис **use:...**, как описано выше. В этом последнем случае, однако, переменная окажется пуста.

### **havebest, havebest\***

Если пакет имеет лучшую стабильную или, соответственно, нестабильную версию, переменная получает значение 1.

### **upgrade, upgradeorinstall, downgrade, recommend, recommendorinstall**

**upgrade** получает значение 1, если пакет установлен и как минимум один слот может быть обновлен (или если лучшая стабильная версия занимает новый слот, а значение переменной **UPDATE\_TO\_HIGHEST\_SLOT** - true). Другие переменные из этого ряда аналогичным образом проверяют, может ли пакет быть обновлен или установлен заново, следует ли откатить его до более ранней версии, может/должно ли быть произведено обновление/откат, может/должно ли быть произведено обновление/откат/установка, соответственно. Переменная **RECOMMEND\_LOCAL\_MODE** определяет, подчиняется ли описанная проверка **LOCAL\_PORTAGE\_CONFIG**.

### **bestupgrade, bestupgradeorinstall, bestdowngrade, bestrecommend, bestrecommendorinstall**

Как в предыдущем случае, с той разницей, что во внимание принимается только лучшая стабильная версия пакета (а не все слоты).

### **better, worse, differ, bestbetter, bestworse, bestdiffer**

Могут использоваться только в условных выражениях внутри **DIFF\_FORMAT\_CHANGED**. **better** получает значение 1, если новый пакет имеет новый слот или новую стабильную версию (или прежнюю версию, но из другого оверлея) в любом слоте. **worse** ведет себя аналогично в том случае, если прежняя версия имеет как минимум один лучший слот или слот, недоступный для нового пакета. **differ** возвращает единицу в том случае, если не все лучшие стабильные слоты прежних и новых пакетов совпадают. Соответствующие **best\***-версии ведут себя аналогично с той разницей, что учитываются только лучшие стабильные версии (а не все слоты). Переменная **RECOMMEND\_LOCAL\_MODE** определяет, подчиняется ли проверка настройкам переменной **LOCAL\_PORTAGE\_CONFIG**.

### **old..., new...**

Могут использоваться только внутри **DIFF\_FORMAT\_CHANGED**. Любое свойство может получить префикс **old** или **new** и соответствующее этому свойству значение, но с учетом, соответственно, прежних или новых данных. Если префикса **old** или **new** нет, по умолчанию принимается новая версия. Например, конструкция **oldavailableversions:ПЕРЕМЕННАЯ** выведет предыдущие доступные версии (**ПЕРЕМЕННАЯ** при этом определяет формат вывода), а **newavailableversions:ПЕРЕМЕННАЯ** и **availableversions:ПЕРЕМЕННАЯ** выведут доступные версии текущего (т.е. нового) пакета.

## **Цвета вывода**

(ИМЯ, ЯРКОСТЬ; МАРКЕРЫ)

Последовательность ",ЯРКОСТЬ" и/или ";МАРКЕРЫ" может быть опущена.

При **ЯРКОСТИ**, равной 1, eix будет использовать для **ИМени** соответствующий 'яркий' (насыщенный) цвет ; если **ЯРКОСТЬ** равна 0 - стандартный цвет.

Пустое **ИМЯ** соответствует значению **default**, и, в отличие от значения **none**, возвращает вывод к стандартным настройкам цвета, без маркеров.

Пустое значение (или отсутствие) **МАРКЕРОВ** соответствует значению **none**, т.е. изменения атрибутов не происходит. Можно одновременно указывать несколько атрибутов маркировки (через запятую).

Доступные цвета: **default** (по умолчанию), **none** (нет), **black** (черный), **red** (красный), **green** (зеленый), **yellow** (желтый), **blue** (синий), **purple** (лиловый), **cyan** (ультрамарин), **gray** (серый).

Доступные атрибуты маркировки: *none* (*нет*), *bold* (*жирный*), *underline* (*подчеркивание*), *blink* (*мигание*), *inverse* (*негатив*).

## Примеры:

```
FORMAT='{installed}(yellow,1;underline){else}(yellow,0){}()\n' eix ...
```

Если пакет ... установлен, выводит его имя ярким желтым цветом с подчеркиванием, иначе обычным желтым цветом.

```
FORMAT='\\n INSTFORMAT='{first}:{}{!last}\\n\\t{}' DATEFORMAT='%x' eix autom*
```

Для каждого пакета **autom\*** выводит имя с указанием категории, а если пакет установлен, то и установленные версии и соответствующие им даты установки. Тот же результат можно было бы получить, вместо `\t\\n{} поставив :{else}\\t\\n{}, поскольку в конце каждой версии, кроме последней, и в начале каждой версии, кроме первой, выводится, разумеется, одна и та же последовательность \t\\n.`

```
FORMAT='{ downgrade } %{FORMAT_ALL}{} eix -I
```

Будут выведены все установленные пакеты, которые рекомендуется откатить до более ранних версий. Заметьте, что конструкция `FORMAT='{ downgrade } %{FORMAT}{}'` не работает, поскольку в этом случае в отложенной замене возникает определение со ссылкой на само себя; естественно, переменную нельзя определять значением самой этой переменной. Поэтому начиная с версии eix 0.13.4 содержимое **FORMAT** и других подобных ей переменных значительно упростилось: `%{FORMAT_ALL}` (для других родственных переменных аналогично). Таким образом, вы можете свободно вставлять полное определение исходного значения **FORMAT** (что мы и сделали в приведенном примере).

Обратите внимание: если вы хотите представить предыдущий пример в компактном выводе, вы не можете просто добавить опцию `-c` – единственным эффектом, которого вы добьетесь, будет то, что для определения строки формата будет использована переменная **FORMAT\_COMPACT**, а не **FORMAT**. Для того, чтобы действительно получить компактный вывод, используйте либо:

```
FORMAT_COMPACT='{ downgrade } %{FORMAT_ALL_COMPACT}{} eix -Ic
```

либо (что еще проще):

```
FORMAT='{ downgrade } %{FORMAT_ALL_COMPACT}{} eix -I
```

## ФАЙЛЫ

### /etc/eix-sync.conf

В данном файле хранятся команды и настройки, применяемые к сценарию **eix-sync**. Комментарии в нем начинаются с символа `#` (строки обрезаются при первом вхождении `#`; с `#` нельзя использовать маски). Строки могут иметь описанный ниже вид и исполняются в заданном порядке перед вызовом **emerge --sync**.

Обратите внимание, что к этому файлу добавляется содержимое переменной **EIX\_SYNC\_CONF**. По умолчанию она расширена на **EIX\_SYNC\_OPTS**, так что значения этой переменной также добавляются в файл. Обе упомянутые переменные могут использоваться для переопределения настроек по умолчанию, указанных в **/etc/eix-sync.conf**. При выполнении **eix-sync** анализируется большая часть этих строк, поэтому необходимо учитывать требования безопасности!

### опция/-ии

Использовать **опцию/-ии** по умолчанию для **eix-sync** (ставится перед всеми остальными опциями). В общем случае **опции** должны начинаться с символа `"-"`. **опции** анализируются оболочкой внутри сценария **eix-sync**, поэтому обратите внимание на безопасность и убедитесь, что разделители команд оболочки правильно оформлены кавычками!

### Имя

Вызвать команду **layman -s Имя**. Утилита **layman** предназначена для синхронизации оверлеев и доступна в портежах как **app-portage/layman**.

\*

Вызвать **layman -S** (т.е. синхронизировать оверлеи с помощью layman).

#### **!команда**

Анализировать команду внутри сценария оболочки **eix-sync** (в той же оболочке). команда должна успешно завершить работу, в противном случае **eix-sync** сделает останов и выдаст ошибку. (Таким образом, конструкция **!layman Имя** практически идентична записи **Имя**.) Если вы хотите, чтобы статус выхода игнорировался, пишите в конце команды "; true".

Вы можете использовать данную функцию для распаковки оверлея перед вызовом **layman** или для внесения локальных изменений после вызова **layman**.

В последних версиях e10h команда уже ничего не выводит (если вам необходим вывод, используйте **einfo**) и не анализируется в подоболочке. Иными словами, при необходимости вы можете свободно редактировать переменные окружения, начинать/завершать перенаправление. Недостаток такого поведения состоит в том, что не менее легко и допустить ошибку: переопределить внутренние переменные или функции **eix-sync**. Если вы опасаетесь что-либо нарушить, заключайте команду в скобки (...), чтобы она запускалась в подоболочке.

#### **!!команда**

Аналогично записи **! команда** с той разницей, что в данном случае команда будет выполнена в любом случае, даже при использовании опций **-d**, **-u**, **-l**. Это позволяет устанавливать переменные окружения для других программ.

#### **~команда**

Имеет смысл только при использовании **eix-sync** опций **-s** и **-2**. В таком случае команда будет выполнена перед первым вызовом **rsync**; вывод команды анализируется внутри оболочки **eix-sync**. Если команда или анализ ее вывода отрабатывают с ошибкой, **eix-sync** перестанет выполнять и выдаст сообщение об ошибке. Это можно использовать, например, чтобы выполнить **keychain** и возвратить содержимое соответствующего файла **~/.keychain/\*-sh**, либо для возвращения команд экспорта для текущих переменных **SSH\_AUTH\_SOCK** и **SSH\_AGENT\_PID**. Допускается также вывод команды в виде команды, изменяющей значения переменных **PORTAGE\_RSYNC\_OPTS**, **PORTAGE\_RSYNC\_EXTRA\_OPTS**, **PORTDIR**, **PORTDIR\_SERVER**, **PORTDIR\_CLIENT**, **SERVER**, **CLIENT**. При вызове команды данные переменные получат значения по умолчанию и в дальнейшем будут использованы для команд **rsync** с их очевидным значением.

#### **@команда**

Добавить привязку к команде; фактическое выполнение команды будет отложено до вызова **emerge --sync** (успешного).

#### **@@команда**

Добавить привязку к команде; фактическое выполнение команды будет отложено до вызова **emerge --sync** (успешного) с последующим **eix-update**.

Вот **примерная** последовательность привязок/команд:

привязки **!!**

**cp /var/cache/eix /var/cache/eix.previous**

вызов **layman**, привязки **!** в порядке, указанном в **/etc/eix-sync.conf**

привязки **~**

**emerge --sync**

привязки **@**

**eix-update**

привязки **@@**

**eix-diff /var/cache/eix.previous**

Несколько полезных примеров вы найдете в **/etc/eix-sync.conf**:

## **-C --ignore-default-opt**

Используйте эту строку, если у вас включена опция --ask в переменной EMERGE\_DEFAULT\_OPTS в файле /etc/make.conf, но вы не хотите, чтобы при выполнении eix-sync было затребовано подтверждение операции.

## **-r -M**

Используйте эти параметры, если у вас **PORTDIR\_CACHE\_METHOD=assign**, а функция **FEATURES=metadata-transfer** неактивна или (по умолчанию в последних версиях portage) отключена. При необходимости вы можете также использовать **-r** и **-M** по отдельности или заменить **-M** строкой.

## **@emerge --regen**

Запускать emerge --regen вместо emerge --metadata.

## **@egencache --repo=local --update**

Обновлять кэш метаданных оверлея с именем репозитария local между вызовами eix-update и emerge --sync. Это имеет смысл в том случае, если для данного оверлея используется метод кэширования **metadata-flat**, и вы хотите убедиться, что кэш метаданных обновлен. Побочным эффектом этого поведения является то, что команда **egencache** будет выполняться даже в том случае, если репозитарий не изменился (так обычно и происходит с локальными репозитариями).

## **!egencache --repo=foo --update**

Обновить кэш метаданных оверлея с именем репозитария foo. Это имеет смысл в том случае, если ранее у вас была строка **foo** или **!команда для обновления foo**, призванная обеспечить обновление репозитария (будь то с помощью layman или какой-либо другой утилиты), а вы хотите использовать для этого оверлея метод кэширования **metadata-flat**, несмотря на то, что кэш метаданных содержит ошибки или не существует в данном репозитарии.

## **!!exec >/var/log/eix-sync.log ; chown portage: /var/log/eix-sync.log || true**

Направить вывод в журнал (с соответствующими правами). true в конце обозначает, что выполнение сценария будет продолжено принудительно даже в том случае, если chown выдает ошибку. Если вы не хотите перенаправлять вывод конечного **eix-diff**, вам следует использовать эту конструкцию в связке со следующей:

## **@@exec >/dev/tty**

Отображать статистику выполнения **eix-diff** в терминале, даже если было сделано перенаправление.

## **@@exit 0**

Не выполнять **eix-diff**.

## **!!export FORCE\_USECOLORS= \${FORCE\_USECOLORS:-true}**

Если только переменная FORCE\_USECOLORS не определяет иное для вашего окружения, вывод eix будет цветным даже в случае перенаправления.

## **~keychain --quiet ~/.ssh/id\_rsa ; cat ~/.keychain/ `hostname`-sh**

## **@@eix-remote update \* `portageq portdir`/local/layman/eix-caches.tar.bz2 \***

(Если вам не нужна локальная копия данных из каталога /usr/portage/local/layman, последний аргумент можно опустить.)

## **@eix-remote fetch /var/cache/remote-cache.tbz2**

## **@@eix-remote add /var/cache/remote-cache.tbz2**

## **/etc/eixrc**

Глобальный конфигурационный файл eix. Значения переменных в **~/.eixrc** или в окружении могут переопределять переменные, определенные в этом файле. См. **~/.eixrc**.

## EIXRC

Если эта переменная окружения установлена, то ее значение будет использовано вместо /etc/eixrc при обращении к данным конфигурации. В таком случае содержимое файла ~/.eixrc будет проигнорировано (но вы, разумеется, можете указать явно, что должны использоваться его значения, если это необходимо).

## EIX\_SYNC\_OPTS, EIX\_SYNC\_CONF, EIX\_REMOTE\_OPTS, EIX\_LAYMAN\_OPTS, EIX\_TEST\_OBSOLETE\_OPTS

Хотя эти переменные обычно настраиваются в ~/.eixrc (описаны в соответствующем разделе), мы упоминаем их здесь, поскольку они чрезвычайно важны для безопасности. Они (по крайней мере некоторые) анализируются оболочкой, если вы запускаете сценарии с соответствующим именем. Поэтому вы непременно должны убедиться, что при запуске этих сценариев из-под root'a переменные не будут изменены.

## ~/.eixrc

Пользовательский файл конфигурации. Переменные в этом файле могут быть переопределены переменными окружения. Для присвоения им значений вы можете использовать shell-подобный синтаксис. Вы можете подключать и другие файлы, использовать вспомогательные переменные для настройки.

Если вы используете вспомогательные переменные обычным образом, вы можете только просматривать заменяемые значения с помощью --dump или --dump-defaults и не можете заменять подставляемые значения, например, в окружении.

Таким образом, вы можете обращаться к переменным не только используя обычный синтаксис оболочки, но и с помощью конструкции `%{ПЕРЕМЕННАЯ}` (скобки можно опускать). Это означает, что замена будет отложена до того момента, когда все файлы конфигурации и переменные среды будут обработаны. Отложенная замена не отображается при использовании опций with --dump или --dump-defaults.

Такое поведение именуется отложенной заменой / отложенным обращением и предоставляет несколько дополнительных возможностей:

### Специальные символы для отложенной замены

#### `%{%`

Требуется в случае, если вам необходимо использовать в переменной `%{` (в противном случае произойдет отложенная замена).

#### `*ПЕРЕМЕННАЯ`

Если отложенное обращение использует имя переменной, начинающееся с `*`, символ `*` заменяется на `EIX_` или `DIFF_`, в зависимости от того, вызывается ли переменная из `eix/eix-update` или из `eix-diff`. Это позволяет иметь различные настройки по умолчанию для этих программ.

Так, отложенное обращение `%{*ПЕРЕМЕННАЯ}` подставит расширенное значение `EIX_ПЕРЕМЕННАЯ` или `DIFF_ПЕРЕМЕННАЯ`, соответственно.

Атрибуты `\` и `*` могут быть совмещены (порядок не имеет значения).

#### `\ПЕРЕМЕННАЯ`

Если отложенное обращение использует имя переменной, начинающееся с `\`, все символы `\` символы `[\\n\\r\\t]` в значении (расширенной) `ПЕРЕМЕННОЙ` рассматриваются как экранированные.

Так, отложенное обращение `%{ПЕРЕМЕННАЯ}` может быть использовано в таких переменных как `CACHE_METHOD` или `EIX_LOCAL_SETS`; это гарантирует, что `ПЕРЕМЕННАЯ` имеет не более одной "необработанной" записи, даже если она содержит пробелы или обратный слэш.

Атрибуты `\` и `*` могут быть совмещены (порядок не имеет значения).

### Условные блоки в отложенных обращениях

Если вы хотите заменять значения нескольких переменных совершенно иным образом, в зависимости от статуса определенной переменной (булевой), вы можете использовать условные блоки.

В известной степени их функционирование здесь аналогично поведению условных блоков в FORMATSTRING: если переменная, к которой происходит обращение, наконец получает булево значение true (**true/1/yes/y/on**) (или, соответственно, любое непустое значение, если **ПЕРЕМЕННАЯ** имеет префиксом дополнительный символ ?), в результате будет возвращено значение true, и соответствующий блок строки будет расширен. Можно использовать отрицание условий, чтобы расширялась часть "else", когда условие истинно, и часть "if", когда условие ложно. Часть "else" может не выполняться вовсе. Допустимо использовать особые имена переменных: \***ПЕРЕМЕННАЯ** вместо **ПЕРЕМЕННАЯ**.

**%{?ПЕРЕМЕННАЯ}TCODE%{}**

Расширить TCODE, если **ПЕРЕМЕННАЯ** получает значение true.

**%{??ПЕРЕМЕННАЯ}TCODE%{}**

Расширить TCODE, если **ПЕРЕМЕННАЯ** получает значением непустую строку.

**%{!ПЕРЕМЕННАЯ}TCODE%{}**

Расширить TCODE, если **ПЕРЕМЕННАЯ** получает значение false.

**%{!?ПЕРЕМЕННАЯ}TCODE%{}**

Расширить TCODE, **ПЕРЕМЕННАЯ** получает значением пустую строку.

**%{?ПЕРЕМЕННАЯ}TCODE%{else}FCODE%{}**

Расширить TCODE, если **ПЕРЕМЕННАЯ** получает значение true, иначе FCODE.

**%{??ПЕРЕМЕННАЯ}TCODE%{else}FCODE%{}**

Расширить TCODE, если **ПЕРЕМЕННАЯ** получает значением непустую строку, иначе FCODE.

**%{!ПЕРЕМЕННАЯ}TCODE%{else}FCODE%{}**

Расширить TCODE, если **ПЕРЕМЕННАЯ** получает значением false, иначе FCODE.

**%{!?ПЕРЕМЕННАЯ}TCODE%{else}FCODE%{}**

Расширить TCODE, если **ПЕРЕМЕННАЯ** получает значением пустую строку, иначе FCODE.

Условный блок может и целиком помещаться в одной переменной. В этом случае будет невозможно, например, заменить символ %{} путем отложенного обращения к другой переменной (но в TCODE/FCODE использование отложенных обращений допускается).

Заметьте, что все переменные, которые вы добавляете для отложенной замены, выводятся с помощью опции --dump, только если они действительно используются (т.е. если к ним обращаются какие-либо другие переменные). Если вы хотите видеть их во всех случаях, например, для удобства комментирования или последующих изменений, вы можете собрать ссылки на них в переменной **DUMMY**.

Следующие переменные не содержат используемых только для отложенных обращений. Чтобы получить подробное описание этих последних переменных (и их стандартных настроек), выполните **eix --dump**.

## **DUMMY (строка)**

Эта переменная не влияет напрямую на программы, но может быть использована для хранения отложенных обращений к переменным (не обрабатываемым иным образом), чтобы их можно было просмотреть с помощью --dump или --dump-defaults.

## **EIX\_SYNC\_OPTS, EIX\_SYNC\_CONF (строка)**

Значение переменной **EIX\_SYNC\_CONF** добавляется в файл `@SYSCONFDIR/eix-sync.conf`. Подробности см. в описании этого файла. По умолчанию **EIX\_SYNC\_CONF** представляет собой отложенную замену для **EIX\_SYNC\_OPTS**, т.е. вы обычно можете использовать **EIX\_SYNC\_OPTS** в тех же целях. Помните, что если одна из этих переменных будет поставлена под угрозу, при вызове **eix-sync** с привилегиями администратора может произойти практически всё что угодно - подумайте о безопасности! Если вы используете эти переменные, аккуратно расставляйте кавычки!

### **EIX\_REMOTE\_OPTS, EIX\_LAYMAN\_OPTS, EIX\_TEST\_OSOLETE\_OPTS (строка)**

Содержимое этих переменных анализируется и используется в качестве аргументов сценариями **eix-remote**, **eix-layman** и **eix-test-obsolete** соответственно. Поэтому не забывайте о рисках безопасности и всегда ставьте переменные в кавычки!

### **EIXRC\_SOURCE (строка)**

Данный путь используется как префикс source-команд в `/etc/eixrc`. Стандартно он должен быть определен в окружении, но может быть определен и в файле `/etc/eixrc`. В последнем случае он переопределит значение переменной окружения непосредственно при обработке, пока всем файлам не будут сопоставлены источники. Обратите внимание, что в момент считывания значения переменной отложенная замена еще не произведена.

### **EIX\_PREFIX (строка префикса)** ("строка префикса" означает строку, но если она имеет значение "/", оно будет заменено на ")

Эта переменная предназначена главным образом для присвоения значения в окружении, если вы хотите использовать chroot. Она применяется как префикс пути, по которому ищется `/etc/eixrc`. Если этот путь не определен, служит префиксом значения переменной окружения **PORTAGE\_CONFIGROOT**. Если не определена и она, используется значение переменной по умолчанию (как правило, пустое).

Кроме того, данная переменная используется при отложенной замене для определения префикса путей других переменных; подробнее о переменных, которые к ней обращаются, см. в выводе **eix --dump**. При текущих настройках по умолчанию она изменяет все пути, за исключением:

### **EPREFIX (строка префикса)**

Для этой переменной значение по умолчанию берется из **EIX\_PREFIX**; к нему добавляется умолчание, зависящее от конфигурации (для prefix-portage). Эта переменная не имеет собственного значения и используется при отложенной замене для определения префикса путей других переменных (если не изменены их значения по умолчанию); подробнее о переменных, которые к ней обращаются, см. в выводе **eix --dump**. При текущих настройках по умолчанию она изменяет все пути, за исключением:

`/usr/bin/eix-functions.sh`

`~/.eixrc`

путь к файлу/-ам кэша, переданный командной строке

**PORTAGE\_PROFILE** (переменная, но не ссылка)

**PORTDIR**

**Пути доступа к оверлеям**

Последние три могут быть изменены через **EPREFIX\_TREE**.

Переменная **EPREFIX** предназначена для обеспечения доступа, подобного chroot, к prefix-portage. Заметьте, что как пользователь prefix-portage вы должны будете выполнять **eix-sync** с опцией **-e**.

### **ROOT (строка префикса)**

**EPREFIX\_TREE (строка префикса)**

**EPREFIX\_ROOT (строка префикса)**

На самом деле это не внутренние переменные eix - они просто используются для отложенной замены со следующими переменными, аналогично **EPREFIX** (см. ниже).

Для базового отслеживания использования portage данной переменной служит переменная **ROOT**. Заметьте, что переменные в файле /etc/make.conf не переопределяют переменные конфигурации eix. Так, команда **ROOT=что-либо**, прописанная в /etc/make.conf, никак не скажется на поведении eix. Если вам нужно обратное, установите новое значение в окружении или в конфигурационном файле eix.

Вы можете легко изменить пути, к которым относятся **EPREFIX** и **ROOT**: для этого просто используйте при определении соответствующих переменных отложенные обращения **%{EPREFIX}**, **%{ROOT}**, **%{EPREFIX\_ROOT}** (которые, в свою очередь, определяются как отложенное обращение, так что вы можете свободно переопределять поведение программы, в случае если и **EPREFIX**, и **ROOT** непусты: вы можете совместить эти два пути, а можете выбрать только один из них), или же не используйте ничего. Разумеется, для отложенного обращения вы можете назначать и другие переменные, например:

**EIX\_CACHEFILE= %{EPREFIX\_PROFILE}/var/cache/eix**

Используйте эту настройку, если вы хотите, чтобы файл кэша eix зависел от профиля администратора (и только от него).

### **PORTAGE\_CONFIGROOT (строка префикса)**

Используется как префикс путей каталога /etc. Задача - сохранять PORTAGE\_CONFIGROOT аналогично тому, как это делает portage. Если вы определяете эту переменную в окружении, одновременно изменится и путь доступа к /etc/eixrc. (Заметьте, что обращение к файлу /etc/eixrc происходит до выполнения отложенной замены.)

### **MAKE\_GLOBALS (строка)**

Если этот файл существует, он будет использован вместо **%{PORTAGE\_CONFIGROOT}/etc/make.globals**. Значение по умолчанию соответствует поведению \*portage-2.2\* и выше.

### **EPREFIX\_SOURCE (строка префикса)**

Используется как префикс путей - аргументов source-команд в файлах /etc/make.conf и /etc/make.globals.

### **EPREFIX\_INSTALLED (строка префикса)**

Используется как префикс пути, по которому eix ожидает информацию об установленных пакетах.

### **EPREFIX\_PORTAGE\_CACHE (строка префикса)**

Используется как префикс пути к кэшу портежей.

### **EPREFIX\_ACCESS\_OVERLAYS (строка префикса)**

Используется как префикс путей к оверлеям, для доступа к их файлам.

### **EPREFIX\_PORTDIR (строка префикса)**

Используется как префикс **PORTDIR**.

### **EPREFIX\_OVERLAYS (строка префикса)**

Используется как префикс всех записей в **PORTAGE\_OVERLAY**.

### **EPREFIX\_PROFILE (строка префикса)**

Используется как префикс **PORTAGE\_PROFILE** (переменной, а не ссылки).

### **EPREFIX\_VIRTUAL (строка префикса)**

Используется как префикс оверлеев в базе данных eix для проверки их существования.

### **EIX\_CACHEFILE (строка)**

Файла кэша eix (обычно **%{EPREFIX}/var/cache/eix**).

### **EIX\_WORLD (строка)**

Файл, который eix рассматривает как файл world. Учтите, что обычно этот файл доступен для чтения только в том случае, если вы входите в группу portage. Во избежание проблем с правами доступа можно использовать переменную **SAVE\_WORLD**.

### **EIX\_WORLD\_SETS (строка)**

Файл, который eix рассматривает как файл world\_sets. Для него справедливо всё сказанное для **EIX\_WORLD**.

### **EIX\_LOCAL\_SETS (список строк)(что такое список строк, объясняется далее)**

Это список каталогов, которые содержат локально определенные сеты. Каталоги считаются в заданном порядке; файлы с именами сетов, которые были обработаны ранее, игнорируются: в этом отношении предшествующие записи в **EIX\_LOCAL\_SETS** имеют приоритет над новыми.

Для относительных каталогов (т.е. каталогов, имена которых не начинаются с символа /) родительским считается **\$PORTDIR**. Записи в **EIX\_LOCAL\_SETS**, начинающиеся со специального символа \*, рассматриваются особо, как несколько записей; при этом символ \* последовательно заменяется путями доступа к оверлеям, в обратном порядке (это сделано потому, что в некотором смысле более ранние записи **EIX\_LOCAL\_SETS** переопределяют более поздние, в то время как для переменной **PORTDIR\_OVERLAY** программа ожидает обратного).

По умолчанию эта переменная содержит **/etc/portage/sets**, **/etc/portage/sets.eix**, **sets** (т.е., в общем,  **\${PORTDIR}/sets**, исходя из предыдущего абзаца), **\*/sets** (т.е., в общем,  **\${PORTDIR\_OVERLAY}/sets**, исходя из предыдущего абзаца), а также **%{EIX\_LOCAL\_SETS\_ADD}**. Последнее обращение позволяет вам добавлять дополнительные каталоги к значению переменной **%{EIX\_LOCAL\_SETS\_ADD}** в файле **/etc/eixrc**. Это может быть полезно, в частности, если вы определили другой каталог для множественных сетов (например, для оверлея или для обработки "world-candidate = False") в файле **/etc/portage/sets.conf** вашей системы или в файле **sets.conf file** какого-либо оверлея.

Во всех списках строк разделителями записей могут служить [ \t\r\n]. Если вы хотите использовать один из них внутри строки, не забудьте экранировать его с помощью \. Все обратные слэши также должны быть экранированы, поскольку служебные обратные слэши при символах обратного слэша и при разделителях удаляются. Если вы хотите использовать переменную в "обычном" виде, вы можете прибегнуть к отложенной замене: конструкция **%{ПЕРЕМЕННАЯ}** вставит в текст значение **ПЕРЕМЕННОЙ** с экранированными [ \t\r\n]. (Подробнее см. в разделе об отложенной замене.)

### **EAPI\_REGEX**

Это регулярное выражение соответствует распознанным EAPI в суффиксах ебилдов, интерпретируемых согласно GLEP 55. Может потребоваться изменить это выражение локально, в зависимости от установленной версии portage (чтобы убедиться, что она умеет производить синтаксический анализ соответствующих EAPI). В исключительных случаях переменная может оказаться пуста; тогда все ебилды с EAPI-суффиксами будут проигнорированы.

### **SAVE\_WORLD (true/false)**

Если эта переменная имеет значение true, данные вашего world-файла будут храниться в **/var/cache/eix**. Таким образом, всякий, кто имеет доступ к этому файлу, будет располагать содержимым вашего "мира". Если вы этого не хотите, не ставьте значением переменной true.

### **CURRENT\_WORLD (true/false)**

При значении false будут использованы данные world-файла, хранящиеся в файле **/var/cache/eix**, даже если текущий world сам доступен для чтения. Иначе эти данные будут переопределены текущим world-файлом (при условии, что он доступен для чтения).

### **SKIP\_PERMISSION\_TESTS (true/false)**

При значении **true** eix-update не будет проверять группы и привилегии. Вероятно, это имеет смысл, если вы используете более тонкий способ настройки прав доступа (NSS/LDAP, acl, утилиты, использующие pam и т.д.), иначе eix может неверно определить права. Единственный недостаток такого режима заключается в том, что eix-update не будет выдавать ошибок в начале, а выдаст ошибку (иногда неверно ее истолковав) при попытке получить доступ к кэш-файлу без достаточных привилегий.

## **EBUILD\_USER, EBUILD\_GROUP, EBUILD\_UID, EBUILD\_GID**

Эти переменные определяют права методов кэширования **ebuild/ebuild\***. Подробнее см. в разделе о данных методах кэширования.

## **PORTRAGE\_ROOTPATH, PORTAGE\_BIN\_PATH**

Если эти переменные не пусты, они без изменений передаются сценарию ebuild.sh, при кэшировании методом **ebuild\***. Заметьте, что ebuild.sh использует данные переменные для расчета пути PATH, так что здесь кроется серьезная угроза безопасности.

## **NOFOUND\_STATUS (целое число)**

Это значение используется в качестве статуса выхода в том случае, если найдено 0 соответствий запросу. Учитывается значение переменной **COUNT\_ONLY\_PRINTED**.

## **MOREFOUND\_STATUS (целое число)**

Это значение используется в качестве статуса выхода в том случае, если найдено хотя бы 2 соответствия запросу. Учитывается значение переменной **COUNT\_ONLY\_PRINTED**.

## **QUICKMODE (true/false)**

Если переменная имеет значение true, eix и eix-diff по умолчанию будут использовать опцию **--quick**.

## **CAREMODE (true/false)**

Если переменная имеет значение true, eix и eix-diff по умолчанию будут использовать опцию **--care**.

## **USE\_BUILD\_TIME (true/false)**

Переменная определяет, использовать ли запись BUILD\_TIME из базы данных портежей (если она существует) вместо отметки времени каталога (обычно это время установки). Разница между двумя этими возможностями имеет значение только для пакетов, устанавливаемых из .tbz2-архивов. В большинстве случаев важнее время сборки; эти данные к тому же надежнее. К сожалению, время сборки доступно только для пакетов, собранных и установленных portage не ниже версии 2.2\_rc63. Чтобы выяснить, к каким пакетам это относится, а к каким нет, можно использовать команду **eix-installed [no-]builddate**. Считывание времени сборки всегда занимает больше времени, чем использование отметки времени из каталога (даже если время сборки недоступно). Поэтому, возможно, будет удобнее назначить данной переменной значение **false**, если для вас важнее скорость работы eix, чем корректность данных времени сборки.

## **QUIETMODE (true/false)**

Если переменная имеет значение true, eix и eix-diff по умолчанию будут использовать опцию **--quiet**.

## **PRINT\_APPEND (строка)**

Эта строка добавляется к выводу опции **--print**. Внутри нее стандартно интерпретируются специальные символы, например, \n. По умолчанию - новая строка для разумного вывода в интерактивной оболочке. Обратите внимание, что при замене команд в сценариях оболочки удаляются все пробелы в конце, поэтому новая строка не создаст здесь никакого неудобства (а пробелы в конце переменной в сценарии в любом случае будут удалены). Поэтому чтобы считывать переменные, предоставляемые eix, внутри сценария оболочки без опущения завершающих пробелов, следует использовать видимый символ для переменной PRINT\_APPEND и работать с конструкциями типа **\*ПЕРЕМЕННАЯ="`PRINT\_APPEND=x eix --print ПЕРЕМЕННАЯ` ; ПЕРЕМЕННАЯ="\${VAR%\$x}"\*  
\*NEWLINE\* \*(true\*/false)\* p(((**). Если переменная имеет значение true, eix будет переходить на новую строку после каждой версии, для которой строка формата предусматривает какой-либо вывод. Значение **\*true\*** может быть полезно, если вы используете пустые строки формата, выводящие только одну строку (если не хотите вручную добавлять переход на новую строку в строке формата). Однако если вы

используете строку формата, которая выводит по версии на строку, причем отображаются только те версии, которые удовлетворяют определенному запросу, оказывается довольно неудобно отслеживать непосредственно в строке формата, делать ли явный переход на новую строку или это произойдет автоматически. В этом случае оптимальным решением будет значение \*NEWLINE=false\*. Стандартная строка формата eix корректно обрабатывает оба варианта (проверяет значение переменной \*NEWLINE\* и переходит на новую строку, только если это требуется). p(((. Если только вам не нужна поддержка строки формата, которую вы прописали ранее, мы рекомендуем сохранить значение по умолчанию \*NEWLINE=false\* и вручную добавить \*\n\* (или явно перейти на новую строку) в конце строки формата. Если же вы самостоятельно писали строки формата, следует обновить их так, чтобы происходил переход на новую строку, а не указывать \*NEWLINE=true\*. Данная возможность существует лишь в интересах обратной совместимости и может быть исключена в следующих версиях eix. \*DEFAULT\_FORMAT\* \*(normal/\*compact/\*verbose)\* p(((. Определяет, какой из двух режимов - \*--compact\* или \*--verbose\* - используется по умолчанию. \*DIFF\_ONLY\_INSTALLED\* \*(true/\*false)\* p(((. Если переменная имеет значение true, eix-diff будет обрабатывать изменения версии только для установленных пакетов. \*DIFF\_NO\_SLOTS\* \*(true/\*false)\* p(((. Если переменная имеет значение true, eix-diff не будет обрабатывать слоты для изменившихся версий. \*DIFF\_SEPARATE\_DELETED\* \*(true/\*false)\* p(((. Если переменная имеет значение true, eix-diff отобразит удаленные пакеты в отдельном блоке. Иначе eix-diff выведет удаленные и измененные пакеты общим алфавитным списком. \*DIFF\_PRINT\_HEADER\* \*(true/\*false)\* p(((. Если переменная имеет значение true, eix-diff выведет информационный заголовок. \*NO\_RESTRICTIONS\* \*(true/\*false)\* p(((. Если переменная имеет значение false, будут выведены данные RESTRICTION и PROPERTIES. \*RESTRICT\_INSTALLED\* \*(true/\*false)\* p(((. Если переменная имеет значение true, будут рассчитываться ограничения загрузки и зеркала для установленных версий. \*CARE\_RESTRICT\_INSTALLED\* \*(true/\*false)\* p(((. Если переменная имеет значение true, ограничения загрузки и зеркала для установленных версий всегда будут браться с диска, даже если эти данные могут быть считаны непосредственно с соответствующей версии пакета. Это медленнее, но надежнее, так как будут выявлены и измененные ограничения. \*FORMAT\*, \*FORMAT\_COMPACT\*, \*FORMAT\_VERBOSE\* \*(строка)\* p(((. Эти переменные определяют соответственно стандартный, компактный и подробный форматы вывода \*eix\*. См. \*FORMATSTRING\*. Начиная с eix-0.13.4 эти переменные просто расширяются на отложенную замену переменных \*FORMAT\_ALL\*, \*FORMAT\_ALL\_COMPACT\* и \*FORMAT\_ALL\_VERBOSE\*, соответственно. Такое определение призвано упростить доступ к стандартному определению, когда вы меняете определение \*FORMAT\*. \*DIFF\_FORMAT\_NEW\*, \*DIFF\_FORMAT\_DELETE\*, \*DIFF\_FORMAT\_CHANGED\* \*(строка)\* p(((. Эти переменные определяют формат вывода для пакетов, которые были добавлены, удалены или для которых стала доступна новая стабильная версия. Они используются только \*eix-diff\*. См. \*FORMATSTRING\*. Начиная с eix-0.13.4 эти переменные просто расширяются на отложенную замену переменных \*DIFF\_FORMAT\_ALL\_NEW\*, \*DIFF\_FORMAT\_ALL\_DELETE\* и \*DIFF\_FORMAT\_ALL\_CHANGED\*, соответственно.

## **FORMAT\_INSTALLATION\_DATE, FORMAT\_SHORT\_INSTALLATION\_DATE**

Определяется формат функции strftime(), используемые для отображения времени установки (в стандартном или, соответственно, сокращенном виде).

## **FORMAT\_INSTALLED\_USE**

Определяет printf-подобный формат, используемый для отображения USE-флагов установленных пакетов. Если значением переменной является пустая строка, обработка происходит несколько быстрее, поскольку данные не будут считываться.

## **FORMAT\_BEFORE\_KEYWORDS, FORMAT\_AFTER\_KEYWORDS, FORMAT\_BEFORE\_EFFECTIVE\_KEYWORDS, FORMAT\_AFTER\_EFFECTIVE\_KEYWORDS**

Эти строки выводятся перед/после строки KEYWORDS (текущей) для данной версии.

## **FORMAT\_BEFORE\_SET\_USE, FORMAT\_AFTER\_SET\_USE, FORMAT\_BEFORE\_UNSET\_USE, FORMAT\_AFTER\_UNSET\_USE**

Эти строки выводятся перед/после включенных/отключенных USE-флагов установленных версий.

## **FORCE\_USECOLORS (true/false)**

Отображать в цвете, даже если стандартный вывод не терминал.

### **FORCE\_PERCENTAGE (true/false)**

Отображать ход выполнения операции в процентах, даже если стандартный вывод не терминал.

### **STYLE\_VERSION\_SORTED (true/false)**

Определяет, включена ли по умолчанию опция **--versionsort**.

### **STYLE\_VERSION\_LINES (true/false)**

Определяет, включена ли по умолчанию опция **--versionlines**.

### **DUP\_PACKAGES\_ONLY\_OVERLAYS (true/false)**

Определяет, производится ли проверка дублирующих друг друга пакетов только среди оверлеев, т.е. будет ли считаться пакет дублированным только в том случае, если он существует по меньшей мере в двух оверлеях.

### **DUP VERSIONS ONLY OVERLAYS (true/false)**

Определяет, производится ли проверка дублирующих друг друга версий только среди оверлеев, т.е. будет ли считаться пакет дублированным только в том случае, если он существует по меньшей мере в двух оверлеях.

### **DEFAULT\_IS\_OR (true/false)**

Если имеется несколько шаблонов в аргументах, не соединенных логически (операторами **-a** и **-o**), eix подразумевает их неявное объединение. Если значением этой переменной является **true**, подразумевается, что производится дизъюнкция **-o** (or), иначе конъюнкция **-a** (and).

### **OVERLAYS\_LIST (all/all-if-used/all-used/all-used-renumbered/no)**

Пользователи, работающие со множеством оверлеев, хотели ли бы, чтобы в конце отображались не все оверлеи, а только действительно используемые. Это поведение настраивается здесь. Значение переменной интерпретируется следующим образом:

#### **all-if-used/if-used/if**

Отображать все оверлеи, если используется хотя бы один из них. В версиях eix ниже 0.6.0 эта настройка предлагалась по умолчанию.

#### **used-renumbered/renumber/renumbered/number**

Отображать только действительно используемые оверлеи с "правильной" нумерацией (т.е. если используется только два оверлея, они будут пронумерованы как [1] и [2]). Этот подход неудобен тем, что нумерация оверлеев меняется в зависимости от запроса. Впрочем, порядок следования остается прежним.

#### **all-used/only-used/used**

Отображать только действительно используемые оверлеи, сохраняя единую, не зависящую от запроса нумерацию (в одной базе данных).

#### **no/false**

Никогда не выводить перечень оверлеев.

иное

Выводить перечень оверлеев при каждом запросе (даже если ни один из них не нужен).

### **LEVENSHTEIN\_DISTANCE (целое число)**

Устанавливает расстояние Левенштейна по умолчанию.

### **EXCLUDE\_OVERLAY (список строк)**

Устанавливает перечень подстановочных шаблонов для путей к оверлеям, исключенным из индекса. См. опцию **--exclude-overlay** для **\*eix-update\***.

## **ADD\_OVERLAY (список строк)**

Устанавливает перечень оверлеев, добавленных в индекс. См. опцию **--add-overlay** для \*eix-update\*.

## **EXPORT\_PORTDIR\_OVERLAY (true/false)**

Если значение переменной true, и все оверлеи исключены/добавлены, будет экспортирована соответственно измененная переменная \*PORTDIR\_OVERLAY\*. Это означает, что в некотором смысле и соответствующие еклассы этих оверлеев исключены/добавлены для методов кэширования eix и eix\*. \*CACHE\_METHOD\_PARSE\* \*(строка)\* p(((. Эта строка добавляется ко всем методам кэширования, использующим parse/parse\*/ebuild/ebuild\*. По умолчанию содержит строку \*#metadata-flat\*. Если только вы не экспериментируете с настройками, вам, скорее всего, потребуется именно этот метод. Он подразумевает, что если для ебилда доступны текущие метаданные, они будут использованы, вместо парсинга/исполнения ебилда. Последние либо не вполне надежны (парсинг) либо занимают долго времени (исполнение), поэтому обращение к метаданным, при условии, что они доступны и обновлены, всегда предпочтительнее. \*PORTDIR\_CACHE\_METHOD\*, \*OVERLAY\_CACHE\_METHOD\* \*(строка)\* p(((. Устанавливает тип кэша, используемого портежами и оверлеями. По умолчанию переменная \*PORTDIR\_CACHE\_METHOD\* имеет значение \_metadata-flat\_, \*OVERLAY\_CACHE\_METHOD\* - \_parse|ebuild\*. p(((. \*Внимание:\* Если вы не вполне доверяете ебилдам в используемых вами оверлеях, лучше указать значение \*OVERLAY\_CACHE\_METHOD=parse\*. p(((. Возможно, вы захотите временно выставить значение \*OVERLAY\_CACHE\_METHOD=eix\*::~\* - как будет разъяснено ниже, в этом случае eix по умолчанию будет брать данные об оверлеях из прежней базы данных eix. p(((. Ниже описаны доступные методы кэширования. Возможно, вы захотите переопределить метод кэширования для отдельных оверлеев. Это можно сделать с помощью следующих переменных: \*CACHE\_METHOD\*, \*OVERRIDE\_CACHE\_METHOD\* \*(список строк)\* p(((. Эти переменные представляют собой списки строк вида "\_оверлей\_ \_метод\_ \_оверлей\_ \_метод\_ ...". Методом кэширования \_оверлея\_ устанавливается следующий за ним \_метод\_; тем самым переопределяются значения по умолчанию переменной \*OVERLAY\_CACHE\_METHOD\* (или \*PORTDIR\_CACHE\_METHOD\* если \_оверлей\_ - каталог \*PORTDIR\*). \_оверлей\_ интерпретируется как подстановочный шаблон, который должен соответствовать \_пути\_ к оверлею (разрешаются символические ссылки) (вы не можете указать здесь имя репозитария, вы должны использовать настоящий \_путь\_). Новые записи переопределяют старые: последняя удовлетворяющая запись имеет приоритет. p(((. Переменные \*CACHE\_METHOD\* и \*OVERRIDE\_CACHE\_METHOD\* отличают тем, что первая применяется незамедлительно, тогда как вторая может быть использована для расширения/переопределения неявных изменений при использовании \*KEEP\_VIRTUALS\* (см. ниже). p(((. Определения переменных \*CACHE\_METHOD\* и \*OVERRIDE\_CACHE\_METHOD\* по умолчанию содержат \*%{ADD\_CACHE\_METHOD}\* и, соответственно, \*%{ADD\_OVERRIDE\_CACHE\_METHOD}\*). Согласно механизму отложенной замены (сущность которого мы раскрываем в другом разделе), это означает, что переменные \*ADD\_CACHE\_METHOD\* и \*ADD\_OVERRIDE\_CACHE\_METHOD\* также могут быть использованы для локального переопределения метода кэширования. Если вы изменяете \*CACHE\_METHOD\* или \*OVERRIDE\_CACHE\_METHOD\* в файле /etc/eixrc, рекомендуется добавить "%{ADD\_CACHE\_METHOD}" или, соответственно, "%{ADD\_OVERRIDE\_CACHE\_METHOD}" (обратите внимание на пробел перед %) в конце измененных определений, чтобы эти переменные продолжали применяться для локального переопределения значений.

Доступны следующие методы кэширования:

### **metadata-flat или metadata-flat:ПУТЬ**

Использовать кэш метаданных, расположенный в дереве портежей (\$PORTDIR/metadata/cache). Это стандартный метод, который работает всегда, когда дерево портежей доступно для чтения при запуске eix-update.

Если вы явно пропишете ПУТЬ, это значение переопределит путь, указанный выше. При этом вы должны использовать полный путь (без префикса). Возможно, вы захотите переопределить ПУТЬ, если вы используете какой-либо пакетный менеджер для генерирования метаданных в соответствующем каталоге.

### **metadata-assign или metadata-assign:ПУТЬ**

Аналогично **metadata-flat** с той разницей, что файлы внутри кэша метаданных ожидаются в "формате присвоения" (ТИП=значение). Это справедливо для ряда оверлеев alt-gentoo.

### sqlite

Это чрезвычайно быстрый метод кэширования; для его использования необходимо, что portage работал с бэкендом sqlite. См. [http://en.gentoo-wiki.com/wiki/Portage\\_SQLite\\_Cache](http://en.gentoo-wiki.com/wiki/Portage_SQLite_Cache) (первоначальная версия этого руководства располагалась на странице [http://gentoo-wiki.com/TIP\\_speed\\_up\\_portage\\_with\\_sqlite](http://gentoo-wiki.com/TIP_speed_up_portage_with_sqlite), которая, возможно, всё еще доступна по адресу [http://gentoo-wiki.info/TIP\\_speed\\_up\\_portage\\_with\\_sqlite](http://gentoo-wiki.info/TIP_speed_up_portage_with_sqlite)). Обратите внимание, что, в отличие от метода кэширования **metadata**, принимаемого по умолчанию, при использовании данного метода перед вызовом **eix-update** вы должны будете выполнить **emerge --metadata**.

Поскольку этот метод поддерживается, только если установлен пакет **sqlite**, он может не поддерживаться "из коробки". Если вы планируете его использовать, вам потребуется установить **eix** с соответствующим USE-флагом (или, при ручной установке, выполнить `./configure --with-sqlite` перед компиляцией).

Что касается остальных методов кэширования, они обрабатывают только категории, подключенные через profile/categories (в дереве портежей или оверлее). Если вам это не подходит, используйте **sqlite\*** (см. ниже).

### sqlite\*

Аналогичен методу кэширования **eix** с той разницей, что в кэш добавляются все категории, найденные в **ФАЙЛе**, даже те, которые не были подключены через profile/categories.

### cdb

Используйте этот метод кэширования, если у вас portage версии ниже 2.1 и модуль cdb из <http://forums.gentoo.org/viewtopic-t-261580.html> в качестве бэкенда для portage (cdb со словарями srickle'd в качестве значений). Заметьте, что, в отличие от стандартного **metadata-\***, при использовании данного метода перед вызовом **eix-update** вы должны будете выполнить **emerge --metadata**.

### flat или flat:ПУТЬ

Этот метод аналогичен **metadata-flat** с той разницей, что метаданные ожидаются в каталоге **ПУТЬ - \$ {PORTAGE\_OR\_OVERLAY\_DIR}**. Если \ПУТЬ\_ опущен, его значением по умолчанию принимается **/var/cache/edb/dep**.

Вы можете использовать этот метод кэширования с portage версий ниже 2.1 и бэкендом по умолчанию.

### assign или assign:ПУТЬ

Этот метод аналогичен **flat** с той разницей, что файлы внутри кэша метаданных ожидаются в "формате присвоения" (ТИП=значение). Это справедливо для portage-2.1 с дефолтным бэкендом.

Если вы используете portage-2.1 и выше с бэкендом по умолчанию, вы можете использовать этот метод, если у вас нет доступа к дереву портежей при выполнении eix-update. Заметьте, что, в отличие от стандартного **metadata-\***, при использовании данного метода перед вызовом **eix-update** вы должны будете выполнить **emerge --metadata**. В таком случае, вы скорее всего обратитесь к соответствующей опции eix-sync.

### repo-flat или repo-flat:ПУТЬ / repo-assign или repo-assign:ПУТЬ

Метод аналогичен **flat/assign** с той разницей, что метаданные ожидаются в каталоге **ПУТЬ/\_\$ {repo\_name}**. Если значение \${repo\_name}\_ пусто, за имя репозитария принимается **x-XXX**, где **XXX** - последний ненулевой компонент пути к каталогу портежей/оверлея. Это поведение соответствует политике именования paludis. Если в результате предлагается ошибочный путь к оверлею, вы можете переопределить его вручную, например, так:

**OVERRIDE\_CACHE\_METHOD='ошибочный\путь\_\*\*metadata-assign:исправленный\путь\_к\_оверлею\_\*\*'**.

**parse[#\_метод\_для\_метаданных]...**

Извлечь информацию из ебилдов, проанализировав их эвристически. Использование данного метода хотя и не ставит под угрозу безопасность, но может стать причиной ряда проблем. Например, если переменные получают значения только в екласах, они будут проигнорированы. Среди проблем, с которыми сопряжено применение данного метода, можно указать отсутствующие данные SLOT для типичных ебилдов категории kde-base или фиксированных номеров версий для кросс-компиляторов gcc. Соответствует методу кэширования **none** в старых версиях eix (ниже 0.11.1).

Возможно добавить одну или более строк вида `#метод\для_метаданных`, где под методом \_для\_метаданных\_ подразумевается один из перечисленных выше методов кэширования (исключая **sqlite** и **cdb**). В этом случае методы кэширования метаданных используются для проверки, содержат ли эти метаданные они более новую информацию по сравнению с ебилдом. Если это так, то вместо данных ебилда будут использоваться эти метаданные (приоритет имеют первые удовлетворяющие запросу метаданные).

Как правило, пользователю нужно именно такое поведение, особенно когда речь идет об оверлеях, поскольку метаданные надежнее, чем результаты, полученные методом кэширования **parse**. Разумеется, это имеет смысл лишь в том случае, если метаданные нашего оверлея *foo* регулярно обновляются мейнтейнерами оверлея.

#### **egencache --repo=foo --update**

(Будет ли выполнено это действие для оверлеев layout, зависит от команды сопровождения оверлея; для своих локальных оверлеев вы, естественно, можете вручную ввести приведенную команду.

Как правило, это поведение удовлетворяет большинство пользователей, поэтому строка в **CACHE\_METHOD\_PARSE** по умолчанию добавляется к методу кэширования **parse**.

Конечно, если вы заранее знаете, что метаданные оверлея свежие, будет (несколько) быстрее напрямую использовать подходящий метод кэширования (как правило, **metadata-flat**). Это имеет смысл, когда, например, вызывается **egencache** из сценария синхронизации оверлея (e.g. in eix-sync).

#### **parse\*[#метод-для\_метаданных]...**

Этот метод кэширования наалогичен **parse** с той разницей, что переменные не расширяются в определениях переменных. Он соответствует методу кэширования **none\*** из предыдущих (ниже 0.11.1) и ранних версий eix (before 0.7.1)

#### **ebuild[#метод\_для\_метаданных]...**

Если кэш портежей недоступен (например, речь идет об оверлеях), этот метод кэширования обеспечивает наибольшую совместимость, но и наиболее медленный. Данные извлекаются из ебилда с помощью `"/usr/bin/ebuild ... depend"`. Поскольку все ебилды будут выполняться в оболочке bash, это может быть небезопасно, если вы не уверены в надежности всех ebuild-сценариев и переменных окружения. **eix-update** предпримет попытку заменить пользователя/группу на **EBUILD\_USER/EBUILD\_GROUP** (если предыдущая переменная не имеет смысла - используя **EBUILD\_UID/EBUILD\_GID**), прежде чем выполнять ебилд. Окружение преднамеренно не инициализируется до фактического начала выполнения действия, чтобы можно было последовательно передать несколько переменных ебилду. Однако это же может привести к неожиданному поведению или даже к угрозе безопасности, поскольку многие сценарии bash могут быть обойдены в разных окружениях. Во избежание проблем используйте **env -i eix-update**, если для вас важен этот аспект.

**Используйте этот метод, только если вы уверены в надежности всех ебилдов, к которым он будет применен!**

Об опциональных дополнениях `#метод\для_метаданных` см. в описании метода кэширования **parse**. Заметьте, что, если доступны соответствующие метаданные, в общем случае этот вариант окажется быстрее, чем выполнение ебилда.

#### **ebuild\*[#метод\_для\_метаданных]...**

Этот метод представляет собой несколько более быструю, хотя и в меньшей степени совместимую версию **ebuild**: данные берутся недокументированного "/usr/lib/portage/ebuild.sh". Вместо выполнения программы на для каждого ебилда, как это происходит при использовании метода кэширования **ebuild**, "only" a lengthy shell-script and the ebuild itself is executed (таким образом, этот метод небезопасен, если вы не вполне уверены в надежности всех ebuild-сценариев). Большинство переменных окружения, за исключением переменных portage и PATH, инициализируются; PORTAGE\_BIN\_PATH и PORTAGE\_ROOTPATH экспортируются (учтите, что ebuild.sh использует эти переменные для вычисления PATH, создавая тем самым серьезную угрозу безопасности); некоторые специфичные для ебилдов переменные, такие как \$P, получают свои значения при выполнении ебилда. Этот метод обеспечивает несколько худшую совместимость, нежели **ebuild**, и успешность его применения в большей степени зависит от версии portage. Тем не менее он работает значительно быстрее, чем **ebuild**, и достаточно стабилен для обработки, например, типичных ебилдов категории kde-base.

**Используйте этот метод, только если вы уверены в надежности всех ебилдов, к которым он будет применен!**

**parse|ebuild, parse\*|ebuild, parse|ebuild\*, parse\*|ebuild\*** [#метод\_для\_метаданных]...

Здесь скомбинированы методы кэширования **parse/parse\*** и **ebuild/ebuild\***. Сначала каждый ебилд проверяется на возможность использования метода **parse/parse\***. Если предполагаемый результат содержит лакуны или выглядит странно, ебилд проверяется на возможность использования метода **ebuild/ebuild\***. По эмпирическим данным этот метод работает значительно быстрее, чем **ebuild/ebuild\***, но всё же намного медленнее, чем **parse/parse\***. Конечно, он имеет те же риски безопасности, что и **ebuild/ebuild\***.

**Используйте этот метод, только если вы уверены в надежности всех ебилдов, к которым он будет применен!**

**eix** или **eix:ФАЙЛ** или **eix:ФАЙЛ:оверлей**

Использовать кэш-файл **ФАЙЛ**, предварительно сгенерированный **eix-update**. Если значение опущено или пусто, по умолчанию **ФАЙЛ** - это /var/cache/eix.

Что касается остальных методов кэширования, для них будут считываться только категории, подключенные через profile/category (в дереве портежей или оверлея). Если вам это не нужно, используйте **eix\*** (см. ниже).

Если **оверлей** не задан или имеет пустое значение, будет обрабатываться только основное дерево портежей в **ФАЙЛе**, а оверлеи из **ФАЙЛА** игнорируются. Иначе будет считываться только та часть **ФАЙЛА**, которая соответствует **оверлею**. Под соответствием мы здесь подразумеваем, что будет использован первый оверлей из **ФАЙЛА**, удовлетворяющий подстановочному шаблону. При проверке, удовлетворяет ли оверлей шаблону, сначала сканируется метка, затем путь и, наконец, номер оверлея в **ФАЙЛЕ**. Заметьте, что **оверлей** обычно не зависит от действительных имен оверлеев или их порядка - роль играют только имена/порядок, сохраненные в **ФАЙЛЕ**.

Существуют два исключения для оверлея, обрабатываемые согласно другим правилам:

Если **оверлей** имеет специальное значение "**~**", в качестве аргумента **\_оверлей\_** будет неявно использована действительная метка этого оверлея; если действительная метка оверлея пуста или не подходит, используется текущий путь к оверлею.

Если **оверлей** имеет специальное значение "**\***", будут обработаны **все** оверлеи в **ФАЙЛЕ**. Для большинства пользователей это предпочтительное поведение, поскольку в результате, если **ФАЙЛ** изначально содержал несколько оверлеев, структура оверлеев будет "выровнена".

**eix\*** или **eix\*:ФАЙЛ** или **eix\*:ФАЙЛ:оверлей**

Аналогичен методу кэширования **eix** с той разницей, что добавляются все категории, найденные в **ФАЙЛе**, даже те, которые были включены через profile/categories.

Данный метод кэширования имеет смысл, если **ФАЙЛ** содержит данные какого-либо каталога оверлея, для которого неизвестны или, возможно, устарели соответствующие файлы profile/categories на локальном хосте.

Применяется для **eix-remote**.

Обратите внимание, что, в частности, при использовании **PORTDIR\_CACHE\_METHOD="eix\*::~"**, данные оверлея по умолчанию лишь "копируются" из предыдущего кэш-файла eix.

### **KEEP\_VIRTUALS (true/false)**

Если переменная имеет значение true, eix-update сохранит все виртуальные оверлеи из предыдущей базы данных, если таковые имеются. Тот же результат вы получили бы, добавив для каждого виртуального оверлея из старой базы "имя-оверлея" в переменной **ADD\_OVERLAY** и соответствующую запись "оверлей eix\*::оверлей" в переменной **CACHE\_METHOD**. Это означает, что данная опция должна переопределять настройки **CACHE\_METHOD**, но может быть сама переопределена значением **OVERRIDE\_CACHE\_METHOD**.

### **REPO\_NAMES**

Эта переменная содержит список строк вида "*шаблон-каталога метка-оверлея шаблон-каталога метка оверлея ...*". При создании нового файла кэша оверлей, совпадающий с шаблоном-каталогом, получает метку-оверлея, независимо от содержимого файла profiles/repo\_name. Эта переменная может также сопоставлять метки виртуальным оверлеям, которые не содержат такого файла. Кроме того, переменная также переопределяет метки оверлеев, установленные переменной **KEEP\_VIRTUALS**. Старые записи переопределяются новыми: приоритет имеет последнее соответствие.

### **LOCAL\_PORTAGE\_CONFIG (true/false)**

Если переменная имеет значение false, содержимое /etc/portage и **ACCEPT\_KEYWORDS** (будь то в файле make.conf или в окружении) игнорируется. В eix начиная с версии 0.7.9 рекомендуется оставлять за этой переменной значение true, поскольку при значении false часть информации будет упущена.

### **ALWAYS\_ACCEPT\_KEYWORDS (true/false)**

Если переменная имеет значение true, переменная **ACCEPT\_KEYWORDS** будет использована даже без **LOCAL\_PORTAGE\_CONFIG**, например, для определения стабильности "по умолчанию".

### **UPGRADE\_LOCAL\_MODE (+ или local/- или non-local/иное)**

Если переменная имеет значение + / -, вызов eix с опцией --upgrade всегда будет производить поиск так, как если бы переменная **LOCAL\_PORTAGE\_CONFIG** имела значение **true / false**.

### **RECOMMEND\_LOCAL\_MODE (+ или local/- или non-local/иное)**

Если переменная имеет значение + / -, рекомендации по обновлению/откату, равно как и проверка изменений версий, производимая eix-diff, всегда будут отрабатывать, как если бы переменная **LOCAL\_PORTAGE\_CONFIG** имела значение **true / false**.

### **UPGRADE\_TO\_HIGHEST\_SLOT (true/false)**

Если переменная имеет значение true, любая проверка предложит на обновление установленный пакет не с лучшей стабильной версией в слоте. Исключения из этого общего правила можно прописать в **/etc/portage/package.slot\_upgrade\_forbid** или, соответственно, **/etc/portage/package.slot\_upgrade\_allow**.

### **RECURSIVE\_SETS (true/false)**

Если переменная имеет значение true, сеты и пакеты в составе включенного сета рассматриваются как часть сета более высокого уровня.

### **PRINT\_EFFECTIVE\_KEYWORDS (true/false)**

Если переменная **PRINT\_KEYWORDS** используется и ее значение равно true, причем ключевое слово KEYWORD версии изменено профилем, то на основе значения KEYWORDS, установленного в ебилде, будут отображены действующие ключевые слова (определяются профилем).

### **XML\_KEYWORDS(full/effective/both/true/full\*/effective\*/none/false)**

При использовании опции --xml данная переменная решает, отображать ли для каждой версии полные/действительные ключевые слова **KEYWORDS** (или оба типа). Под "полным" (full) ключевым словом здесь подразумевается строка **KEYWORDS** в ебилде, а под "действительным" (effective) - ключевое слово, трансформированное профилем. Значения **full\*/effective\*** аналогичны **full/effective**, но для обоих типов ключевых слов вывод будет произведен только в том случае, если их значения различаются. Значения **true/false** эквивалентны **full\*/none**.

### **XML\_OVERLAY (true/false)**

При использовании опции --xml эта переменная определяет, будет ли для каждой версии отображаться оверлей (т.е. путь к нему). Эта настройка не повлияет на версии из оверлеев без метки (имени репозитария): для таких версий вывод во всех случаях подразумевает отображение пути доступа к оверлею.

### **SORT\_INST\_USE\_ALPHA (true/false)**

Если переменная имеет значение **true**, USE-флаги установленных пакетов будут выведены в алфавитном порядке. Иначе первыми отображаются (в алфавитном порядке) те флаги, статус которых был изменен при установке пакета, а затем (также в алфавитном порядке) все остальные флаги.

### **CHECK\_INSTALLED\_OVERLAYS (true/false/repository)**

Если переменная имеет значение **true**, утилита всегда будет проверять, из какого оверлея установленный пакет. Если ее значение **false**, будут проверены только пакеты с версиями, доступными как минимум в двух деревьях - иными словами, только те пакеты, которые, судя по базе данных, могли быть установлены из другого оверлея. Однако это может быть и не так, если такой пакет был удален из какого-либо оверлея или если сам оверлей теперь отсутствует в базе данных.

Вы можете использовать специальное, компромиссное значение **repository**: если при установке были сохранены данные о репозитарии (в последних версиях portage это именно так; чтобы проверить, как ведет себя ваша версия portage, выполните **eix-installed [no-]repository**), то во всех случаях будут использоваться эти данные. Только если данные нечитаемы или не соответствуют состоянию системы, поведение утилиты для данной версии будет таким же, как при **CHECK\_INSTALLED\_OVERLAYS=false**.

Если вы выставите значением этой переменной **false** или **repository**, особенно в сочетании с опциями -T (если значение **NONEEXISTENT\_IF\_OTHER\_OVERLAY** равно **true**) и -J, то получите огромный прирост скорости. Но вы должны отдавать себе отчет в том, что используемые в этом случае сведения об установленном оверлее не вполне надежны (для версий, установленных с ранними версиями portage). В частности, опция -T не обнаружит, что установленная версия пакета происходит из дублированного оверлея, если в текущей базе данных все версии пакета из одного (другого) дерева.

### **OBSOLETE\_MINUSASTERISK (true/false)**

Если переменная имеет значение **true**, обрабатывать -\* в /etc/portage/package.keywords так, как это делал portage версии 2.1.2 и ниже. Уже в portage-2.1.2 ключевое слово -\* фактически устарело и было заменено на \*\*, которому может соответствовать всё что угодно (даже ебилд с пустым значением **KEYWORDS**). Заметьте, что существуют родственные ключевые слова \* и ~\*, которые разрешают установку стабильных или, соответственно, нестабильных версий любой архитектуры, указанной в **KEYWORDS**.

### **PRINT\_COUNT\_ALWAYS (true/false/never)**

Если переменная имеет значение **true**, в последней строке всегда будет отображаться количество совпадений с запросом, даже если их 0 или 1. Если **PRINT\_COUNT\_ALWAYS=never**, то последняя строка в любом случае будет опущена. В обычных условиях неудобен и тот, и другой вариант, однако они могут упростить написание отдельных сценариев, основанных на парсинге вывода eix.

### **COUNT\_ONLY\_PRINTED (true/false)**

Если переменная имеет значение `false`, будет выведено только количество соответствий запросу, независимо от того, насколько они результативны в действительности. Это может быть полезно для ускорения отдельных сценариев, для которых вам важно только число соответствий, если вы используете, например, `FORMAT=`.

### **DEFAULT\_MATCH\_FIELD (список строк)**

Это список строк вида `регулярное\соответствие[\n\r\t ]поле_соответствия, который используется для определения стандартного поля соответствия. Шаблон поиска в командной строке сопоставляется со всеми регулярными_соответствиями из этого списка в заданном порядке. Для первого сопоставления по умолчанию используется соответствующее поле_соответствия. Вы можете указать последнее поле_соответствия (без регулярного_выражения) в этом списке - оно используется как резервное в случае, если соответствий не было обнаружено; если такой записи не существует, по умолчанию принимается name. Чтобы дополнительно не экранировать символы вручную, может быть удобно использовать отложенную замену  ${\\\!ПЕРЕМЕННАЯ} для регулярного_выражения_.`

Для поля\соответствия\_ допустимы значения: `name, category, category/name` (или `category-name`), `description, license, homepage, provide, virtual, set, slot, installed-slot, use` (или `iuse`), `with-use` (или `installed-with-use`), `without-use` (или `installed-without-use`). Они соответствуют аналогичным консольным опциям для поля соответствия. Специальное значение `virtual` совмещает в себе две опции командной строки - `-A` и `-P`.

### **DEFAULT\_MATCH\_ALGORITHM (список строк)**

Это список строк вида `регулярное\выражение[\n\r\t ]алгоритм_соответствия_, который используется для определения стандартного алгоритма соответствия. Аналогично DEFAULT_MATCH_FIELD с тем отличием, что алгоритм\соответствия указывает на предпочтительный алгоритм соответствия. алгоритм_соответствия_ может принимать значения regex, pattern, substring, begin, end, exact, fuzzy. Они соответствуют аналогичным консольным опциям для алгоритма соответствия. Если не указано другого алгоритма соответствия по умолчанию, будет использоваться regex.`

### **TEST\_FOR\_EMPTY (true/false)**

Определяет, отображать ли пустые записи в `/etc/portage/package.*` с опцией `-t`.

### **TEST\_KEYWORDS (true/false)**

Определяет, сканировать ли `/etc/portage/package.keywords` с помощью опции `-t`.

### **TEST\_MASK (true/false)**

Определяет, сканировать ли `/etc/portage/package.mask` с помощью опции `-t`.

### **TEST\_UNMASK (true/false)**

Определяет, сканировать ли `/etc/portage/package.unmask` с помощью опции `-t`.

### **TEST\_USE (true/false)**

Определяет, сканировать ли `/etc/portage/package.use` с помощью опции `-t`.

### **TEST\_ENV (true/false)**

Определяет, сканировать ли `/etc/portage/package.env` с помощью опции `-t`.

### **TEST\_CFLAGS (true/false)**

Определяет, сканировать ли `/etc/portage/package.cflags` с помощью опции `-t`.

### **TEST\_REMOVED (true/false)**

Определяет, сканировать ли удаленные пакеты с помощью опции `-t`.

### **TEST\_FOR\_NONEXISTENT (true/false)**

Определяет, считать ли несуществующие установленные версии удовлетворяющими проверке с помощью опции `-T`. Определение несуществующих версий содержится в переменных `NONEXISTENT_IF`.

## **TEST\_FOR\_REDUNDANCY (true/false)**

Определяет, считать ли повторяющиеся записи в /etc/portage/package.\* удовлетворяющими проверке с помощью опции -T. Определение повторяющихся записей содержится в переменных **REDUNDANT\_IF**.

## **ACCEPT\_KEYWORDS\_AS\_ARCH (full/true/false)**

Если переменная имеет значение full или true, значение ARCH будет изменено через ACCEPT\_KEYWORDS. Эта переменная определяет, какие ключевые слова считать ARCH и OTHERARCH. Значение full изменяет также исходные ключевые слова ARCH.

## **NONEXISTENT\_IF\_OTHER\_OVERLAY (true/false)**

Определяет, считать ли версии несуществующими для TEST\_FOR\_NONEXISTENT, если они из другого оверлея, нежели установленная версия.

## **NONEXISTENT\_IF\_MASKED (true/false)**

Определяет, считать ли замаскированные версии несуществующими для TEST\_FOR\_NONEXISTENT.

## **REDUNDANT\_IF\_DOUBLE (строка)**

Справедливо, если /etc/portage/package.keywords содержит повторы ключевых слов для некоторых/всех ((не)установленных) версий.

**строка** описывает версии, которые будут подвергнуты обработке. Она может иметь следующие значения:

### **no** или **false**

Не проверять на подобную избыточность.

### **some**

Сигнализировать об избыточности некоторых версий в базе данных.

### **all**

Сигнализировать только об избыточности всех версий в базе данных.

### **some-installed**

Сигнализировать об избыточности некоторых установленных версий в базе данных. Неустановленные версии будут проигнорированы.

### **all-installed**

Сигнализировать только об избыточности всех установленных версий в базе данных. Если версия не установлена, она должна быть указана хотя бы один раз.

### **some-uninstalled**

Сигнализировать об избыточности некоторых неустановленных версий в базе данных. Установленные версии будут проигнорированы.

### **all-uninstalled**

Сигнализировать только об избыточности всех неустановленных версий в базе данных. Если версия установлена, она должна быть указана хотя бы один раз.

**- один\из\_перечисленных\_вариантов или + один\_из\_перечисленных\_вариантов\_**

Проверка имеет смысл, только если дополнительно не установлено ни одной версии пакета (для -) или, соответственно, установлена по крайней мере одна версия (для +).

**один\из\_перечисленных\_вариантов or один\_из\_перечисленных\_вариантов\_**

Результат общей проверки считается положительным, если результативна хотя бы одна частная проверка. Вместо оператора "**or**" дизъюнкция может записываться как "**|**" или "**||**".

## **REDUNDANT\_IF\_DOUBLE\_LINE (строка)**

Справедливо, если /etc/portage/package.keywords содержит две одинаковых строки для одного адресата; в этом случае portage должен отбросить первую из этих двух строк. Заметьте, что строки, адресованные **foo/bar** и **=foo/bar-1**, portage (а вслед за ним eix) не будет считать совпадающими, даже если **foo/bar** относится к версии 1. Для обнаружения повтора в этом последнем, неявном случае можно использовать **REDUNDANT\_IF\_DOUBLE\_LINE**, **REDUNDANT\_IF\_MIXED** и **REDUNDANT\_IF\_STRANGE**.

#### **REDUNDANT\_IF\_MIXED** (string, см. ниже)

Справедливо, если /etc/portage/package.keywords содержит два различных ключевых слова, например, **~ARCH** и **-\***, для релевантных версий.

#### **REDUNDANT\_IF\_WEAKER** (строка, см. ниже)

Справедливо, если /etc/portage/package.keywords для релевантных версий содержит ключевое слово, которое может быть заменено менее жестким ключевым словом - например, **-\*** или **~OTHERARCH** или **OTHERARCH** вместо **~ARCH**, или **~OTHERARCH** вместо **OTHERARCH**.

#### **REDUNDANT\_IF\_STRANGE** (строка, см. ниже)

Справедливо, если /etc/portage/package.keywords для релевантных версий содержит постороннее ключевое слово, например, **UNKNOWNARCH** (неизвестно .ebuild-сценарию и ARCH) или **-OTHERARCH**.

#### **REDUNDANT\_IF\_MINUSASTERISK** (строка, см. ниже)

Справедливо, если /etc/portage/package.keywords содержит запись **-\***. Эта проверка имеет смысл, только если переменная **OBSOLETE\_MINUSASTERISK** имеет значение **false**.

#### **REDUNDANT\_IF\_NO\_CHANGE** (строка, см. ниже)

Справедливо, если /etc/portage/package.keywords предоставляет ключевые слова, которые не изменяют состояние ключевых слов доступности для релевантных версий.

#### **REDUNDANT\_IF\_MASK\_NO\_CHANGE** (строка, см. ниже)

Справедливо, если /etc/portage/package.mask содержит записи, которые не меняют статус маскировки для релевантных версий.

#### **REDUNDANT\_IF\_UNMASK\_NO\_CHANGE** (строка, см. ниже)

Справедливо, если /etc/portage/package.unmask содержит записи, которые не меняют статус маскировки для релевантных версий.

#### **REDUNDANT\_IF\_DOUBLE\_MASKED** (строка, см. ниже)

Справедливо, если /etc/portage/package.mask содержит две одинаковых записи для релевантных версий.

#### **REDUNDANT\_IF\_DOUBLE\_UNMASKED** (строка, см. ниже)

Справедливо, если /etc/portage/package.unmask содержит две одинаковых записи для релевантных версий.

#### **REDUNDANT\_IF\_DOUBLE\_USE** (строка, см. ниже)

Справедливо, если /etc/portage/package.use содержит две одинаковых записи для релевантных версий.

#### **REDUNDANT\_IF\_DOUBLE\_ENV** (строка, см. ниже)

Справедливо, если /etc/portage/package.env содержит две одинаковых записи для релевантных версий.

#### **REDUNDANT\_IF\_DOUBLE\_CFLAGS** (строка, см. ниже)

Справедливо, если /etc/portage/package.cflags содержит две одинаковых записи для релевантных версий. Обратите внимание, что этот файл не поддерживается portage, но вы, возможно, обеспечили его поддержку в вашей личной конфигурации /etc/portage/bashrc. Естественно, это означает и то, что формат для /etc/portage/package.cflags не определен. eix подразумевает, что формат его аналогичен /etc/portage/package.{keywords,use} (т.е. по одной записи на строку, версии, отвечающие критерию, в начале). Как и другие файлы /etc/portage/package.\* , /etc/portage/package.cflags может представлять собой и каталог; в таком случае все находящиеся в нем нескрытые файлы/подкаталоги обрабатываются рекурсивно, разрешаются символические ссылки.

#### **REDUNDANT\_IF\_IN\_KEYWORDS** (строка, см. ниже)

Справедливо, если /etc/portage/package.keywords содержит непустую запись для релевантной версии (для обнаружения пустых записей используйте -t). Конечно, никому не придет в голову видеть избыточность во всех совпадениях (хотя возможно использовать эту опцию нерационально, просто для отображения всех совпадений). Однако совпадающие, но не установленные пакеты могут рассматриваться как избыточные. Поэтому этой переменной, как правило, присваивают значение **-some** или аналогичное ему **-some-uninstalled** (либо **false**, если записи для неустановленных пакетов воспринимать как нормальное явление и не считать избыточными).

#### **REDUNDANT\_IF\_IN\_MASK** (строка, см. ниже)

Аналогично **REDUNDANT\_IF\_IN\_KEYWORDS**, но для /etc/portage/package.mask.

#### **REDUNDANT\_IF\_IN\_UNMASK** (строка, см. ниже)

Аналогично **REDUNDANT\_IF\_IN\_KEYWORDS**, но для /etc/portage/package.unmask.

#### **REDUNDANT\_IF\_IN\_USE** (строка, см. ниже)

Аналогично **REDUNDANT\_IF\_IN\_KEYWORDS**, но для /etc/portage/package.use.

#### **REDUNDANT\_IF\_IN\_ENV** (строка, см. ниже)

Аналогично **REDUNDANT\_IF\_IN\_KEYWORDS**, но для /etc/portage/package.env.

#### **REDUNDANT\_IF\_IN\_CFLAGS** (строка, см. ниже)

Аналогично **REDUNDANT\_IF\_IN\_KEYWORDS**, но для /etc/portage/package.cflags. См. выше комментарии к этому файлу.

#### **SLOT\_UPGRADE\_FORBID** (список строк)

Перечень имен файлов/каталогов, которые используются как /etc/portage/package.slot\_upgrade\_forbid

#### **SLOT\_UPGRADE\_ALLOW** (список строк)

Перечень имен файлов/каталогов, которые используются как /etc/portage/package.slot\_upgrade\_allow

#### **KEYWORDS\_NONEXISTENT** (список строк)

Перечень имен файлов/каталогов, которые используются как /etc/portage/package.accept\_keywords.nonexistent

#### **MASK\_NONEXISTENT** (список строк)

Перечень имен файлов/каталогов, которые используются как /etc/portage/package.mask.nonexistent

#### **UNMASK\_NONEXISTENT** (список строк)

Перечень имен файлов/каталогов, которые используются как /etc/portage/package.unmask.nonexistent

#### **USE\_NONEXISTENT** (список строк)

Перечень имен файлов/каталогов, которые используются как /etc/portage/package.use.nonexistent

#### **ENV\_NONEXISTENT** (список строк)

Перечень имен файлов/каталогов, которые используются как /etc/portage/package.env.nonexistent

#### **CFLAGS\_NONEXISTENT** (список строк)

Перечень имен файлов/каталогов, которые используются как /etc/portage/package.cflags.nonexistent

## **INSTALLED\_NONEXISTENT (список строк)**

Перечень имен файлов/каталогов, которые используются как /etc/portage/package.installed.nonexistent

## **PACKAGE\_NOWARN (список строк)**

Перечень имен файлов/каталогов, которые используются как /etc/portage/package.nowarn

## **/etc/portage/sets.eix**

Каталог, аналогичный /etc/portage/sets (см. справочную страницу portage). Поскольку portage располагает несколькими способами определения сетов пакетов, недоступными eix, вы можете использовать этот каталог для хранения сетов (статических), чтобы eix мог их опознать (чтобы записи для сетов в /etc/portage/package.keywords корректно обрабатывались).

## **/etc/portage/package.slot\_upgrade\_forbid**

## **/etc/portage/package.slot\_upgrade\_allow**

Аналогично другим адресам /etc/portage/package.\*., могут быть как файлами, так и каталогами. Записи в них имеют вид *категория/пакет* (по записи на строку). Соответствующие записям пакеты рассматриваются как исключения для переменной UPGRADE\_TO\_HIGHEST\_SLOT.

## **/etc/portage/package.accept\_keywords.nonexistent**

## **/etc/portage/package.keywords.nonexistent**

## **/etc/portage/package.mask.nonexistent**

## **/etc/portage/package.unmask.nonexistent**

## **/etc/portage/package.use.nonexistent**

## **/etc/portage/package.env.nonexistent**

## **/etc/portage/package.cflags.nonexistent**

Аналогично другим адресам /etc/portage/package.\*., могут быть как файлами, так и каталогами.

Разделителем записей служит либо пробел, либо новая строка. Если какая-либо запись совпадает с первым словом в строке соответствующего файла /etc/portage/package.

{keywords,mask,unmask,use,cflags}, данная строка не будет проверяться опцией -t (на предмет имен, отсутствующих в базе данных). Так вы можете избежать некоторых предупреждений при использовании -t.

## **/etc/portage/package.installed.nonexistent**

Аналогичен другим файлам/каталогам /etc/portage/package.\*.nonexistent с той разницей, что не будут выводится сообщения опции -t об установленных пакетах, которые были удалены из базы данных. Записи в этом файле имеют вид *категория/пакет*, но категорию можно и опустить (хотя это не рекомендуется).

## **/etc/portage/package.nowarn**

Аналогично другим адресам /etc/portage/package.\*., может быть как файлом, так и каталогом. Позволяет отключать проверку -T для отдельных пакетов. Формат файла совпадает с форматом /etc/portage/package.use с той разницей, что существует возможность включения/отключения проверки. Учитываться будут те строки, для которых имеется хотя бы одно соответствие. Так, если он содержит строки:

**sys-kernel/\*-sources no\_change weaker**

**>sys-kernel/hardened-sources-2.6.40 -weaker**

опция -T не обнаружит пакет sys-kernel/\*-sources только потому, что активен ключ **REDUNDANT\_IF\_NO\_CHANGE** или **REDUNDANT\_IF\_WEAKER**. Исключение из правила делается для **REDUNDANT\_IF\_WEAKER** и для hardened-sources только в том случае, если доступная версия пакета на ниже 2.6.40. Записи в файле могут располагаться в произвольном порядке; в случае вхождения символа "-" он всегда имеет приоритет.

В данном файле вы можете указывать один пакет несколько раз. В этом случае будут проверяться вместе все условия, определенные для одного пакета.

Доступные критерии проверки: **in\_keywords**, **no\_change**, **double**, **mixed**, **weaker**, **minusasterisk**, **double\_line**, **in\_mask**, **mask\_no\_change**, **double\_masked**, **in\_unmask**, **unmask\_no\_change**, **double\_unmasked**, **in\_use**, **double\_use**, **in\_env**, **double\_env**, **in\_cflags**, **double\_cflags**. По смыслу они идентичны соответствующим переменным **REDUNDANT\_IF\_\***.

Дополнительно можно использовать критерии **nonexistent**, **masked**, **other\_overlay**, по смыслу идентичные, соответственно, **TEST\_FOR\_NONEXISTENT**, **NONEXISTENT\_IF\_MASKED**, **NONEXISTENT\_IF\_OTHER\_OVERLAY**.

## /var/cache/eix

Это бинарная база данных eix. При необходимости адрес можно изменить с помощью переменной **EIX\_CACHEFILE** (по умолчанию она соотносится с **EPREFIX**, используя механизм отложенного обращения). Но для обеспечения проверки отдельных привилегий перед выполнением **eix-update** необходимо использовать стандартный путь /var/cache/eix.

## versionsort

**versionsort** представляет собой вспомогательный инструмент для обработки сценариев и имеет двоякое назначение. Сначала он отсекает информацию о версии от аргументов либо интерпретирует сами аргументы как строки версий (в зависимости от используемой эвристической процедуры). Затем он отображает собранные строки версий упорядоченным списком, согласно правилам сортировки версий, принятым для portage. Если аргументов несколько, каждая версия (включая последнюю) завершается переходом на новую строку. Так, команда

```
versionsort gcc-4.4 4.4_alpha0 sys-devel/gcc-4.05 4.5
```

выведет следующее:

4.05

4.4\_alpha0

4.4

4.5

Если вы передаете утилите только один аргумент, eix менее жестко подходит к обработке версий: даже если строка версии не вполне корректна, выводимые строки гарантированно будут соответствовать обработанным строкам версий. Вы можете включать в сценарии запуск versionsort с одним аргументом, чтобы отделить имя пакета от версии, например

```
split=1-font-adobe-75dpi-1.3-r1
```

```
version=`versionsort "X${split}"` ; name=${split%"-${version}"}'
```

Хотя в настоящее время разницы нет, надежнее в таком случае (принимая во внимание, что в будущем возможно расширение формата версии), если аргумент(ы) начинается/-ются не с числа (X в приведенном примере), дабы гарантировать, что versionsort действительно отделяет версию от аргумента, а не принимает за версию весь аргумент целиком.

## ОШИБКИ (+ ЧАВО)

Об обнаруженных ошибках либо сообщайте напрямую девелоперам eix:

<http://developer.berlios.de/projects/eix/>, либо оставьте заявку на багтрекере Gentoo: <http://bugs.gentoo.org/>.

eix не поддерживает и, вероятно, никогда не будет поддерживать обработку зависимостей и/или USE-флагов. Это, в частности, означает, что вывод eix -i в целом будет отличаться от вывода emerge update; для нормального функционирования системы ориентироваться следует на последний. Прежде всего это справедливо для обновления пакетов со слотами. Переменная **UPGRADE\_TO\_HIGHEST\_SLOT** и исключения, вручную установленные в **/etc/portage/package.slot\_upgrade\_forbid** либо, соответственно, в **/etc/portage/package.slot\_upgrade\_allow**, могут помочь частично обойти эти неудобства.

eix не обеспечивает и планирует обеспечивать полную поддержку всех поддерживаемых portage сетов. В настоящее время не предполагается, что eix когда либо будет поддерживать даже рекомендуемые настройки PROPERTIES=set для помещения пакетов в дерево (поскольку для их успешной обработки утилите потребовалась бы полная поддержка зависимостей и USE-флагов). В качестве временного решения вы можете вручную определить дополнительные сети в **/etc/portage/sets.eix**. Далее, eix не поддерживает и, вероятно, не будет поддерживать обращение к файлам sets.conf. Если вы указывали дополнительные каталоги **sets/**, например, в каком-либо оверле, вы должны добавить эти каталоги, вручную изменив значение переменной **EIX\_LOCAL\_SETS** в файле **/etc/eixrc**. Проще всего будет поместить в **/etc/eixrc** запись вида **EIX\_LOCAL\_SETS\_ADD="/путь/к/оверлею1/sets /путь/к/оверлею2/sets ..."**

(см. выше описание **EIX\_LOCAL\_SETS**).

eix-diff никогда не обращается к **/etc/portage/profile**. (Это объясняется тем, что сохраненная база данных содержит только сведения о маскировке в исходном профиле, но не сам профиль. С другой стороны, **/etc/portage/profile** может быть интерпретирован, только если профиль известен.)

Если вы хотите использовать номер оверлея в какой-либо переменной/команде с аргументами eix, вывод с применяемым по умолчанию **OVERLAYS\_LIST=all-used-renumbered** неудобен.

Не существует настройки метода кэширования, которая позволяла бы получать информацию из оверлеев (для таковых метаданные кэша портежей недоступны) одновременно с высокой скоростью и надежностью - вам всегда приходится выбирать между ними. Стандартно система предпочитает режим максимальной скорости, но при этом зачастую некорректно отображаются слоты и возникают другие проблемы.

Всегда непросто работать с **EPREFIX/ROOT**. В частности, уже в силу поддержки множества таких переменных, eix не может не быть подвержен локальным атакам, если его вызывать из потенциально небезопасного окружения.

Для многих было неудобно принятное ранее по умолчанию значение **KEEP\_VIRTUALS=true**. Однако с новым, отрицательным значением, сомнительно, что кто-либо вообще обнаружит такую возможность. :(

Для переменной **OBSOLETE\_MINUSASTERISK** не существует значения по умолчанию, которое удовлетворяло бы пользователей как старых, так и новых версий portage.

Eix обладает настолько широким функционалом, что документация по нему и, соответственно, настройка с течением времени чрезвычайно усложнились. И тем не менее многие параметры по-прежнему не поддаются конфигурированию...

## ИСТОРИЯ

Изначально утилита **eix** существовала под названием **portagedb**. Переименована она была во избежание двусмысленности - в составе portage уже имелся инструмент **portagedb**.

В первых версиях функционал eix-update обеспечивался вызовом eix с ключом -i. Затем для облегчения сопровождения он был отделен, и появилась **update-eix**. В конечном итоге исполняемый файл был сохранен, его назначение определяется командой вызова, а оригинальное название, update-eix, превратилось в **eix-update**.

Утилита **eix-diff** сначала также именовалась иначе, а именно **diff-eix**. Нынешняя **eix-remote** была известна как **update-eix-remote**, **eix-layman** - как **update-eix-layman**, а **eix-functions.sh** - как **functions-eix.sh** (а в изначальной версии - как **update-eix-functions.sh**). Все эти переименования преследовали цель создать логично организованный набор утилит: все программы, идущие с eix, теперь имеют префикс **eix-\*** (за исключением **versionsort**, которая, в общем-то, представляет собой самостоятельный

инструмент). Если вам непривычна эта новая схема именования или если вы используете сценарии, зависящие от прежних имен переменных, вы можете использовать символические ссылки на оригинальные имена; такое поведение полностью поддерживается и не будет отменено. Если вы вызываете eix' **./configure** с опцией **--enable-obsolete-symlinks** или **--enable-obsolete-reminder**, такие ссылки (или обертки, которые также напоминают об устаревших именах) создаются автоматически - хотя и не по умолчанию, ведь новым пользователям эта возможность не нужна. Мы рекомендуем обновить ваши сценарии, заменив в них имена обрабатываемых переменных на новые, чтобы их можно было выполнять и на свежих системах. Кроме того, в **eix-functions.sh** по соображениям логики были переименованы две переменные (**CallUpdateEix** -> **CallUpdate** и **ClearUpdateEixArgs** -> **ClearUpdateArgs**); была переименована и внутренняя переменная, к которой они обращаются (но ее никогда не предполагалось использовать в сценариях). Если вы всё еще используете какую-либо из перечисленных опций **./configure**, будет установлена также обертка для **functions-eix.sh**, поддерживающая устаревшие имена функций (и выдающая предупреждение, когда они используются с включенной опцией **--enable-obsolete-reminder**). Начиная с eix-0.8.0, где был введен синтаксис % {\*VARIABLE}, нет смысла использовать разные имена переменных для eix и eix-diff. Поэтому соответствующие **DIFF\_\*** -переменные исчезли.

Метод кэширования **metadata-flat** ранее назывался **metadata**. Метод кэширования **assign** ранее назывался **backport** или **portage-2.1**. Метод кэширования **flat** носил имя **portage-2.0**, его и предпочитали употреблять. Несмотря ни на что, устаревшие имена продолжают поддерживаться.

portage-2.1 и portage-2.1.1 не удаляют старого кэша зависимостей, поэтому при использовании метода кэширования **flat/assign** eix может обнаружить пакеты, которые уже не в дереве портей. Чтобы обойти это поведение, eix-sync ранее удалял сохраненный прежде кэш (rm -rf /var/cache/edb/dep/\*). Поскольку большинство пользователей уже не нуждаются в данном методе кэширования, а удаление старого кэша замедлит следующий запуск portage, оно было исключено из стандартных настроек (но по-прежнему доступно как опция, которую можно прописать в /etc/eix-sync.conf).

Прежде eix-sync по умолчанию работал с gensync instead, а не с layman. Если вы хотите использовать gensync, несмотря на то, что он устарел, см. описание /etc/eix-sync.conf.

eix-sync более не поддерживает журналирование; опции **-v** и **-V** были удалены. Это позволяет избежать таких проблем как отсутствие видимого вывода при выставленном **EMERGE\_DEFAULT\_OPTS=--ask**. Сейчас, если вы хотите вызывать eix-sync по расписанию, используйте перенаправление.

Механизм, который теперь описывается переменной **DEFAULT\_MATCH\_FIELD**, изменился. Использовавшиеся ранее переменные **MATCH\_\*\_IF** и **MATCH\_ORDER** (обладавшие меньшими возможностями) теперь не поддерживаются.

Переменные **ADD\_CACHE\_METHOD** и **ADD\_OVERRIDE\_CACHE\_METHOD** более не являются встроенными, а используются, если не указано иное, при отложенной замене при стандартных **CACHE\_METHOD** и **override\_CACHE\_METHOD**. В частности, настройка переменных в файле /etc/eixrc без добавления отложенной замены "%{ADD\_CACHE\_METHOD}" или, соответственно, "%{ADD\_OVERRIDE\_CACHE\_METHOD}", лишает смысла применяющиеся ранее переменные.

В версиях eix ниже 0.18.0, такие параметры пользовательского отображения версий как **<installedversions:ПЕРЕМЕННАЯ>** или **<availableversions:ПЕРЕМЕННАЯ>** не существовали. Выполнение сопоставленных им функций возлагалось на ряд разрозненных параметров для и набор вариантов вывода списка доступных или установленных версий для вызова из сценариев, подобных . Все они были упразднены, а на смену им пришли переменные **NAMEVERSIONS**, **EQNAMEVERSION**, **ANAMESLOT**, **ANAMEASLOT**, **NAMESLOT**, **NAMEASLOT** и **DATESORT** (причем **DATESORT** предоставляет сравнительный пример, ранее недоступный). См. описание этих переменных в выводе команды **eix --dump**. Чтобы испытать эффект от их использования, попробуйте, например, выполнить:

**eix --format " --pure-packages gcc**

См. также комментарии к опции **-I**.

Начиная с версии eix 0.20.0, логические связки *ВЫРАЖЕНИЙ* претерпели серьезные изменения: теперь они не только допускают скобки, но и цепочки с операторами **-a** и **-o** интуитивно имеют левую ассоциативность. Отрицание **--not** теперь обрабатывается как логическая операция, начинающая новый **ФИГУРНАЯ\_СКОБКА\_ИЛИ\_ПРОВЕРКА**, и не рассматривается как часть **КРИТЕРИЕВ\_ПРОВЕРКИ** (многим пользователям было трудно это воспринимать). Теперь **--pipe** действительно входит в состав **КРИТЕРИЕВ\_ПРОВЕРКИ** и не подразумевает использования логических связок. Начиная с eix-0.20.1, все ранее использовавшиеся файлы /etc/portage/package.\*.nowarn уже не поддерживаются по умолчанию: все их заменил один файл file/dir /etc/portage/package.nowarn. Если вы хотите продолжать использовать множественные файлы .nowarn, установите значение переменной окружения **OBSOLETE\_NOWARN=true**. Подробнее см. в описании переменной **PACKAGE\_NOWARN** (в выводе eix --dump).

В версия eix ниже 0.22.4 **eix-installed** входил в состав **eix-test-obsolete**, что было неудобно и для пользователей, и для обеспечения сопровождения, поскольку эти две утилиты имеют совершенно различное назначение.

## АВТОРЫ

- Martin V\(:ath [\[vaeth@mathematik.uni-wuerzburg.de\]](mailto:vaeth@mathematik.uni-wuerzburg.de)(mailto:vaeth@mathematik.uni-wuerzburg.de)\) (разработчик, текущее сопровождение)
- Emil Beinroth [\[emilbeinroth@gmx.net\]](mailto:emilbeinroth@gmx.net)(mailto:emilbeinroth@gmx.net)\) (разработчик, ранее обеспечивал сопровождение)
- Wolfgang Frisch [\[xororand@users.sourceforge.net\]](mailto:xororand@users.sourceforge.net)(mailto:xororand@users.sourceforge.net)\) (неактивный разработчик, автор первой версии eix)
- Roland Wittmann [\[linuxcommando@users.sourceforge.net\]](mailto:linuxcommando@users.sourceforge.net)(mailto:linuxcommando@users.sourceforge.net)\) (неактивный разработчик)

## СМ. ТАКЖЕ

[portage](#)(5), [fnmatch](#)(3), [regex](#)(7), [emerge](#)(1), [esearch](#)(1), [qsearch](#)(1), [layman](#)(8)

На домашней странице eix, <http://eix.berlios.de/>, содержится дополнительная информация и ссылки.

## ПЕРЕВОД

- Елена Гаврилова [\[e.vl.gavrilova@yandex.ru\]](mailto:e.vl.gavrilova@yandex.ru)(mailto:e.vl.gavrilova@yandex.ru)\)

23 марта 2011

# LAYMAN

## НАЗВАНИЕ

layman - утилита для управления оверлеями Gentoo

## СИНТАКСИС

**layman** [-a] | [--add] [ALL] | [оверлей]

**layman** [-d] | [--delete] [ALL] | [оверлей]

**layman** [-s] | [--sync] [ALL] | [оверлей]

**layman** [-i] | [--info] [ALL] | [оверлей]

**layman** [-S] | [--sync-all]

**layman** [-L] | [--list]

**layman** [-l] | [--list-local]

**layman** [-f] | [--fetch]

# ОПИСАНИЕ

**layman** представляет собой сценарий, позволяющий вам добавлять, удалять и обновлять оверлеи Gentoo из самых различных источников.

## ВАЖНО

С помощью **layman** легко подключать и обновлять оверлеи Gentoo - но, увы, не сложнее и СЛОМАТЬ вашу систему.

В главном дереве портежей содержатся ебилды, за качество которых Gentoo отвечает, поскольку они все поддерживаются разработчиками проекта. Но большинство оверлеев, которые вы можете подключить с помощью **layman**, не дают таких гарантий, поэтому, используя их, вы подвергаете свою систему опасности. Всегда помните об этом, устанавливая сторонние ебилды.

Чтобы обеспечить безопасность системы, вам ОБЯЗАТЕЛЬНО следует ознакомиться с ебилдом, прежде чем его устанавливать.

## ОПЦИИ

### Действия

Ниже перечислены доступные действия **layman**.

#### **-f, --fetch**

Загружает удаленный список оверлеев. Как правило, вам НЕ нужно явно указывать эту опцию: загрузка будет произведена автоматически при запуске действий sync, sync-all или list. Чтобы отказаться от автоматической загрузки, используйте опцию --nofetch.

#### **-a оверлей, --add оверлей**

Добавляет указанный оверлей из кэшированного удаленного списка к вашим локально установленным оверлеям. Чтобы добавить все оверлеи из списка, используйте параметр "ALL".

#### **-d оверлей, --delete оверлей**

Удаляет указанный оверлей из числа установленных локально в вашей системе. Чтобы удалить все оверлеи, используйте параметр "ALL".

#### **-s оверлей, --sync оверлей**

Обновляет указанный оверлей. Чтобы синхронизировать все оверлеи, используйте параметр "ALL".

#### **-i оверлей, --info оверлей**

Выводит всю доступную информацию по указанному оверлею.

#### **-S, --sync-all**

Обновляет все оверлеи. Алиас для -s ALL.

#### **-L, --list**

Выводит содержимое удаленного списка оверлеев.

#### **-l, --list-local**

Выводит список оверлеев, установленных локально.

### Другие опции

Ниже перечислены другие опции **layman**.

#### **-c путь, --config путь**

Устанавливает путь доступа к альтернативному конфигурационному файлу.

#### **-o url, --overlays url**

Устанавливает адреса, по которым расположены дополнительные списки оверлеев. Вы можете использовать этот флаг неоднократно, и указанные URL будут временно добавлены к списку URL в конфигурационном файле. Можно указать и локальные URL, поставив в начале пути **file://**. Однако они будут действительны только для данного запуска layman; чтобы добавить URL на постоянной основе, редактируйте конфигурационный файл. Данная опция полезна при тестировании.

#### **-n, --nofetch**

Не позволяет **layman** автоматически загружать списки удаленных оверлеев. По умолчанию **layman** при каждом действии sync, list или fetch загружает все удаленно размещенные списки.

#### **-k, --nocheck**

Не позволяет **layman** проверять удаленные списки оверлеев на наличие полного описания. По умолчанию **layman** отклоняет оверлеи, для которых не имеется описания или контактной информации.

#### **-q, --quiet**

Полностью отключает вывод сообщений в ходе работы **layman**. В таком "тихом" режиме дочерние процессы запускаются с отключенным стандартным вводом данных во избежание бесконечных интерактивных сессий; таким образом, дочерний процесс может быть прерван в любой ситуации, требующей явных действий пользователя. Это может произойти, например, если у вас оверлей на сервере Subversion и сертификат SSL требует подтверждения вручную.

#### **-v, --verbose**

Задает для **layman** подробный вывод: вам будет предоставлена информация об оверлеях, которые вы можете загрузить.

#### **-N, --nocolor**

Отменяет цветной вывод для **layman**.

#### **-Q УРОВЕНЬ, --quietness УРОВЕНЬ**

Задает для **layman** компактный вывод. Допустимые значения уровня лежат в интервале от 0 до 4, причем 0 соответствует полному отключению информационных сообщений. При уровне ниже 3 справедливы те же ограничения, что для опции **--quiet**.

#### **-p УРОВЕНЬ, --priority УРОВЕНЬ**

Используйте эту опцию в связке с **--add**. Она позволяет изменить приоритет добавленного оверлея и тем самым влияет на порядок записей файла make.conf. Чем ниже выставленный уровень, тем раньше запись появится в списке. Допустимый интервал значений - от 0 до 100. По умолчанию значение принимается равным 50.

## **НАСТРОЙКА**

По умолчанию **layman** считывает параметры конфигурации из файла /etc/layman/layman.cfg. Всего могут быть настроены семь параметров.

### **storage**

Каталог, который будет использован для хранения оверлеев и всех дополнительных данных, необходимых **layman**. По умолчанию это /var/lib/layman. layman сохраняет свои данные не в /var, а в подкаталогах /usr/portage. Было принято решение реализовать поддержку сетевых файловых систем. Если дерево портажей у вас на nfs или подобной файловой системе, и к одному репозитарию ебилдов по сети имеют доступ несколько машин, потребуется также сохранять все необходимые **layman** данные в дереве. Следовательно, и синхронизация оверлеев должна происходить в одном месте.

### **cache**

Здесь **layman** сохраняет загруженный глобальный список оверлеев. Каталог по умолчанию - % (storage)s/cache.xml.

### **overlays**

Здесь **layman** сохраняет список установленных оверлеев. Каталог по умолчанию - %  
(storage)s/overlays.xml.

### **make.conf**

Это конфигурационный файл portage, который **layman** изменяет таким образом, чтобы portage мог обращаться к доступным оверлеям. Путь по умолчанию - %(storage)s/make.conf; можно указывать его и явно, как /etc/make.conf. Но это будет означать, что у вас есть внешняя программа, которая пытается автоматически назначить значения переменным внутри этого важнейшего конфигурационного файла. Мы полагаем, что это небезопасно, и рекомендуем иметь внешний файл минимального размера, определяющий только значение переменной PORTAGE\_OVERLAYS, обращение к которому прописывается в файле /etc/make.conf. Вот почему после установки **layman** предлагает выполнить "echo "source /var/lib/layman/make.conf" >> /etc/make.conf".

### **overlays**

Позволяет указать адрес URL, по которому расположен удаленный список всех доступных оверлеев; по умолчанию это <http://www.gentoo.org/proj/en/overlays/repositories.xml>. Вы можете указать здесь несколько URL (по одному адресу в строке): они будут добавлены к общему списку оверлеев. Таким образом вы можете использовать свою личную коллекцию оверлеев, отсутствующих в глобальном списке.

### **proxy**

Если вам необходимо использовать прокси-сервер, укажите его здесь.

### **nocheck**

Установите значение "yes", если следует запретить **layman** отслеживать оверлеи, не имеющие контактной информации или описания.

## **РАБОТА С ОВЕРЛЕЯМИ**

Утилита **layman** призвана обеспечить удобство сопровождения оверлеев Gentoo без дополнительной настройки.

### **Списки оверлеев**

**layman** позволяет вам загрузить любой оверлей, не изменяя конфигурационных файлов. Для этого выполняемому сценарию необходим внешний список возможных источников данных оверлеев. По адресу <http://www.gentoo.org/proj/en/overlays/repositories.xml> доступен централизованный список, но ничто не мешает вам использовать или выложить для загрузки ваш собственный список оверлеев. Каталог для загрузки удаленных списков может быть также изменен с помощью опции --overlays при запуске **layman**.

Если вы хотите, чтобы к основному списку для **layman** добавили новый оверлей, направьте запрос по адресу [overlays@gentoo.org](mailto:overlays@gentoo.org). Разработчики Gentoo могут сами добавлять свои оверлеи к списку; доступ к нему возможен через репозитарий CVS для веб-сайта проекта Gentoo.

Вы можете также использовать несколько списков одновременно. Для этого допишите нужные вам URL в значение переменной оверлеев в вашем конфигурационном файле, по одному адресу в строке. Тогда **layman** установит содержимое всех списков.

Кроме того, **layman** допускает определение в этом списке локальных файлов. Убедитесь, что пути доступа к ним имеют стандартный для URL префикс file://.

Если для доступа в Интернет вам нужен прокси-сервер, вы можете воспользоваться соответствующей переменной в конфигурационном файле **layman**. Layman также принимает во внимание значение переменной окружения http\_proxy, если оно определено.

### **Локальный кэш**

**layman** сохраняет локальную копию загруженного удаленного списка. По умолчанию она помещается в файл /var/lib/layman/cache.xml. Этот файл существует только в одном экземпляре и перезаписывается каждый раз, когда вы запускаете **layman**.

## Обработка файла /etc/make.conf

Поскольку **layman** призван автоматически включать оверлеи в вашу систему, ему должно быть доступно изменение переменной **PORTDIR\_OVERLAY** в файле /etc/make.conf. Но именно /etc/make.conf является основным конфигурационным файлом Gentoo, жизненно важным для нормального функционирования системы, и его автоматическое изменение было бы опасно. Тем не менее вы можете разрешить это **layman**, установив /etc/make.conf значением переменной **make\_conf** в конфигурационном файле.

Мы рекомендуем другое, значительно более безопасное решение: разрешить **layman** обращаться к внешнему файлу, содержащему лишь переменную **PORTDIR\_OVERLAY** и вызываемому из стандартного /etc/make.conf. Для этого допишите в файле /etc/make.conf следующую строчку:

```
source /var/lib/layman/make.conf
```

Стандартные настройки **layman** содержатся в файле /var/lib/layman/make.conf. Если вы хотите хранить их в другом месте, измените имя этого файла.

При первом обращении файл может не существовать. В этом случае **layman** сам создаст его.

Нет необходимости удалять стандартную переменную **PORTDIR\_OVERLAY** из файла make.conf. По мере необходимости **layman** будет добавлять новые оверлеи к существующему значению, так что все ваши прежние записи сохранятся.

## Добавление, удаление и обновление оверлеев

После загрузки удаленного списка **layman** делает возможным добавление оверлеев из этого списка в вашу систему. Будет предпринята попытка загрузить оверлей. Если загрузка прошла успешно, информация об оверлее будет скопирована из кэша в список локально установленных оверлеев. Кроме того, **layman** при этом изменит значение переменной **PORTDIR\_OVERLAY** таким образом, чтобы оно включало новые пути доступа к оверлеям.

Удаление оверлеев **layman** производит полностью, не оставляя никакого мусора.

Чтобы обновить все оверлеи под управлением **layman**, вы можете запустить его с опцией **--sync ALL** или включив **--sync-all**.

## Вывод списка оверлеев

Чтобы вывести список доступных или установленных оверлеев, запустите **layman** с опциями **--list** или **--list-local** соответственно.

Перед именем каждого полностью поддерживаемого оверлея в списке будет стоять зеленый астериск, а перед именем оверлея, который вам не удастся использовать, поскольку вы не располагаете необходимым функционалом - красный астериск.

По умолчанию **layman** отображает только полностью поддерживаемые оверлеи, опуская остальные. Кроме того, начиная с версии 1.0.7 **layman** выводит предупреждения об оверлеях с пустым полем описания или без контактной информации; если вы предпочитаете поведение **layman**, предлагавшееся в ранних версиях, используйте флаг опции **k** или выставьте опцию **nocheck** в конфигурационном файле.

## Поиск ебилдов в оверлеях

Вы можете искать ебилды, доступные в оверлеях на <http://overlays.gentoo.org>, с помощью утилиты "eix". Установите утилиту, если ее еще нет в вашей системе, и запустите **eix-remote update**.

## Типы оверлеев

В настоящее время **layman** поддерживает оверлеи, экспортируемые через **rsync**, **subversion**, **bzr**, **darcs**, **git**, **mercurial** или распространяемые в виде **tar**-пакетов.

# **СПИСКИ ОВЕРЛЕЕВ**

## **Формат списка оверлеев**

**layman** использует централизованный список оверлеев в формате XML. Этот файл выглядит следующим образом:

### **Локальное добавление оверлея**

Просто создайте список оверлеев в формате, описанном выше, и запустите **layman** с ключом **-o**. Перед локальным URL файла следует поставить **file://**.

### **Глобальное добавление оверлея**

Глобальный список оверлеев, к которому обращается **layman**, находится по адресу <http://www.gentoo.org/proj/en/overlays/repositories.xml>.

Все разработчики Gentoo имеют доступ к этому хранилищу через CVS и могут изменять состав предлагаемых оверлеев.

Если вы не участвуете в разработке Gentoo, но хотите, чтобы ваш оверлей попал в основной список, вам следует связаться с разработчиками проекта оверлеев по адресу [overlays@gentoo.org](mailto:overlays@gentoo.org). Кроме того, вы можете обратиться на канал **#gentoo-overlays** на сервере irc.freenode.net.

## **ПРИМЕРЫ**

### **Установить оверлей**

**layman -f -a wrobel**

К вашему списку установленных оверлеев будет добавлен оверлей **wrobel**.

### **Синхронизировать ваши оверлеи**

**layman -s ALL**

Все оверлеи будут обновлены.

### **Выполнить ряд действий**

**layman -f -a wrobel -a webapps-experimental**

Будет получен удаленный список оверлеев и, незамедлительно после этого, добавлены два указанных оверлея.

## **ФАЙЛЫ**

/etc/layman/layman.cfg

Конфигурационный файл, содержащий настройки по умолчанию для **layman**

## **БАГТРЕКЕР**

Об обнаруженных ошибках сообщайте на <http://bugs.gentoo.org>

## **АВТОР**

- Gunnar Wrobel [\(mailto:wrobel@gentoo.org\)](mailto:wrobel@gentoo.org)

Создатель утилиты.

## **ПРАВА**

Copyright (c) 2005-2009 Gunnar Wrobel

Это свободное программное обеспечение. Вы можете распространять его на условиях GNU GPL v2 (<http://www.gnu.org/licenses/old-licenses/gpl-2.0.html>).

# ПЕРЕВОД

- Елена Гаврилова [\[e.vl.gavrilova@yandex.ru\]\(mailto:e.vl.gavrilova@yandex.ru\)\](mailto:e.vl.gavrilova@yandex.ru)

Июль 2010

## 10. Руководства

1. [Описание IRC](#)
2. [Git распределённая система управления версиями файлов, правка последнего коммита](#)
3. [Руководство по оформлению программ на Python](#)
4. [Программный Raid](#)
5. [Руководство по Iptables](#)
6. [Перекодировка mp3 тегов](#)

### Описание IRC

IRC (англ. Internet Relay Chat --- ретранслируемый интернет-чат) --- сервисная система, при помощи которой можно общаться через сеть Интернет с другими людьми в режиме реального времени.

### Использование

#### Регистрация никна

Регистрация никна происходит следующим образом:

- с помощью команды `/nick` устанавливается ник:

```
/nick supernick
```

- Если выводится сообщение *Nickname is already in use.* или *This nickname is registered.*, это значит, что данный ник уже занят и надо подобрать другой.
- После того, как вы выбрали имя, необходимо начать регистрацию при помощи команды `/msg NickServ REGISTER <пароль> <почта>`. Обязательно укажите реальный почтовый адрес, поскольку на него придут данные, необходимые для завершения регистрации. Вот пример регистрации текущего никна с паролем *mysuperpass* и почтовым адресом *test@post.ru*:

```
/msg NickServ REGISTER mysuperpass test@post.ru
```

- Затем необходимо дождаться письма от irc-сервера, в котором должны присутствовать строки, похожие на эти:

In order to complete your registration, you must send the following command on IRC:

```
/msg NickServ VERIFY REGISTER supernick pbhdoztzxzmzj
```

- Вводим высланную команду для подтверждения регистрации.
- Включаем защиту от несанкционированного использования вашего никна (при попытке использовать этот ник будет дано 30 секунд для ввода пароля):

```
/msg NickServ SET ENFORCE ON
```

### Основы работы с Git

- [Основы работы с Git](#)
- [Введение](#)
- [Основы работы с удаленным репозиторием](#)
- [git clone --- создание копии \(удаленного\) репозитория](#)

- [git fetch и git pull --- забираем изменения из центрального репозитория](#)
- [git push --- вносим изменения в удаленный репозиторий](#)
- [Работа с локальным репозиторием](#)
- [Базовые команды](#)
- [Ветвление](#)
- [Прочие команды и необходимые возможности](#)
- [Серверные команды репозитория](#)
- [Рецепты](#)
- [Ссылки](#)

## **Введение**

**Git** (произн. «гит») - распределённая система управления версиями файлов. Проект был создан Линусом Торвальдсом для управления разработкой ядра Linux. На сегодняшний день поддерживается Джунио Хамано.

Система спроектирована как набор программ, специально разработанных с учётом их использования в скриптах. Это позволяет удобно создавать специализированные системы контроля версий на базе Git или пользовательские интерфейсы. Например, Cogito является именно таким примером фронтенда к репозиториям Git, а StGit использует Git для управления коллекцией патчей.

Git поддерживает быстрое разделение и слияние версий, включает инструменты для визуализации и навигации по нелинейной истории разработки. Как и Darcs, BitKeeper, Mercurial, SVK, Bazaar и Monotone, Git предоставляет каждому разработчику локальную копию всей истории разработки; изменения копируются из одного репозитория в другой.

Удалённый доступ к репозиториям Git обеспечивается git-daemon, gitosis, SSH- или HTTP-сервером. TCP-сервис git-daemon входит в дистрибутив Git и является наряду с SSH наиболее распространённым и надёжным методом доступа. Метод доступа по HTTP, несмотря на ряд ограничений, очень популярен в контролируемых сетях, потому что позволяет использовать существующие конфигурации сетевых фильтров.

## **Основы работы с удаленным репозиторием**

### **git clone --- создание копии (удаленного) репозитория**

Для начала работы с центральным репозиторием, следует создать копию оригинального проекта со всей его историей локально.

Клонируем репозиторий, используя протокол http:

```
git clone http://user@somehost:port/~user/repository/project.git
```

Клонируем репозиторий с той же машины в директорию myrepo:

```
git clone /home/username/project myrepo
```

Клонируем репозиторий, используя безопасный протокол ssh:

```
git clone ssh://user@somehost:port/~user/repository
```

У git имеется и собственный протокол:

```
git clone git://user@somehost:port/~user/repository/project.git/
```

Импортируем svn репозиторий, используя протокол http:

```
git svn clone -s http://repo/location
```

-s -- понимать стандартные папки SVN (trunk, branches, tags)

## **git fetch и git pull --- забираем изменения из центрального репозитория**

Для синхронизации текущей ветки с репозиторием используются команды git fetch и git pull.

git fetch --- забрать изменения удаленной ветки из репозитория по умолчанию, основной ветки; той, которая была использована при клонировании репозитория. Изменения обновят удаленную ветку (remote tracking branch), после чего надо будет провести слияние с локальной веткой командой git merge.

git fetch /home/username/project --- забрать изменения из определенного репозитория.

Возможно также использовать синонимы для адресов, создаваемые командой git remote:

```
git remote add username-project /home/username/project
```

git fetch username-project --- забрать изменения по адресу, определяемому синонимом.

Естественно, что после оценки изменений, например, командой git diff, надо создать коммит слияния с основной:

```
git merge username-project/master
```

Команда git pull сразу забирает изменения и проводит слияние с активной веткой.

Забрать из репозитория, для которого были созданы удаленные ветки по умолчанию:

```
git pull
```

Забрать изменения и метки из определенного репозитория:

```
git pull username-project --tags
```

Как правило, используется сразу команда git pull.

## **git push --- вносим изменения в удаленный репозиторий**

После проведения работы в экспериментальной ветке, слияния с основной, необходимо обновить удаленный репозиторий (удаленную ветку). Для этого используется команда git push.

Отправить свои изменения в удаленную ветку, созданную при клонировании по умолчанию:

```
git push
```

Отправить изменения из ветки master в ветку experimental удаленного репозитория:

```
git push ssh://yourserver.com/~you/proj.git master:experimental
```

В удаленном репозитории origin удалить ветку experimental:

```
git push origin :experimental
```

В удаленную ветку master репозитория origin (синоним репозитория по умолчанию) ветки локальной ветки master:

```
git push origin master:master
```

Отправить метки в удаленную ветку master репозитория origin:

```
git push origin master --tags
```

Изменить указатель для удаленной ветки master репозитория origin (master будет такой же как и develop)

```
git push origin origin/develop:master
```

Добавить ветку test в удаленный репозиторий origin, указывающую на коммит ветки develop:

```
git push origin origin/develop:refs/heads/test
```

# Работа с локальным репозиторием

## Базовые команды

`git init` --- создание репозитория

Команда `git init` создает в директории пустой репозиторий в виде директории `.git`, где и будет в дальнейшем храниться вся информация об истории коммитов, тегах --- о ходе разработки проекта:

```
mkdir project-dir
cd project-dir
git init
```

`git add` и `git rm` --- индексация изменений

Следующее, что нужно знать --- команда `git add`. Она позволяет внести в индекс --- временное хранилище --- изменения, которые затем войдут в коммит. Примеры использования:

индексация измененного файла, либо оповещение о создании нового:

```
git add EDITEDFILE
```

внести в индекс все изменения, включая новые файлы:

```
git add .
```

Из индекса и дерева проекта одновременно файл можно удалить командой `git rm`:

отдельные файлы:

```
git rm FILE1 FILE2
```

хороший пример удаления из документации к `git`, удаляются сразу все файлы `.txt` из папки:

```
git rm Documentation/*.*.txt
```

внести в индекс все удаленные файлы:

```
git rm -r --cached .
```

Сбросить весь индекс или удалить из него изменения определенного файла можно командой `git reset`:

сбросить весь индекс:

```
git reset
```

удалить из индекса конкретный файл:

```
git reset -- EDITEDFILE
```

Команда `git reset` используется не только для сбрасывания индекса, поэтому дальше ей будет уделено гораздо больше внимания.

`git status` --- состояние проекта, измененные и не добавленные файлы, индексированные файлы

Команду `git status`, пожалуй, можно считать самой часто используемой наряду с командами коммита и индексации. Она выводит информацию обо всех изменениях, внесенных в дерево директорий проекта по сравнению с последним коммитом рабочей ветки; отдельно выводятся внесенные в индекс и неиндексированные файлы. Использовать ее крайне просто:

```
git status
```

Кроме того, `git status` указывает на файлы с неразрешенными конфликтами слияния и файлы, игнорируемые `git`.

### `git commit --- совершение коммита`

Коммит --- базовое понятие во всех системах контроля версий, поэтому совершаться он должен легко и по возможности быстро. В простейшем случае достаточно после индексации набрать:

```
git commit
```

Если индекс не пустой, то на его основе будет совершен коммит, после чего пользователя попросят прокомментировать вносимые изменения вызовом команды `edit`. Сохраняемся, и вуаля! Коммит готов.

Есть несколько ключей, упрощающих работу с `git commit`:

```
git commit -a
```

совершит коммит, автоматически индексируя изменения в файлах проекта. Новые файлы при этом индексироваться не будут! Удаление же файлов будет учтено.

```
git commit -m «commit comment»
```

комментируем коммит прямо из командной строки вместо текстового редактора.

```
git commit FILENAME
```

внесет в индекс и создаст коммит на основе изменений единственного файла.

`git reset --- возврат к определенному коммиту, откат изменений, «жесткий» или «мягкий»`

Помимо работы с индексом (см. выше), `git reset` позволяет сбросить состояние проекта до какого-либо коммита в истории. В `git` данное действие может быть двух видов: «мягкого»(soft reset) и «жесткого» (hard reset).

«Мягкий» (с ключом `--soft`) резет оставит нетронутыми ваши индекс и все дерево файлов и директорий проекта, вернется к работе с указанным коммитом. Иными словами, если вы обнаруживаете ошибку в только что совершенном коммите или комментарии к нему, то легко можно исправить ситуацию:

1. `git commit --- некорректный коммит`
  2. `git reset --soft HEAD^ ---` переходим к работе над уже совершенным коммитом, сохраняя все состояние проекта и проиндексированные файлы
  3. `edit WRONGFILE`
  4. `edit ANOTHERWRONGFILE`
  5. `git add .`
  6. `git commit -c ORIG_HEAD ---` вернуться к последнему коммиту, будет предложено редактировать его сообщение. Если сообщение оставить прежним, то достаточно изменить регистр ключа `-c`:
- ```
git commit -C ORIG_HEAD
```

Обратите внимание на обозначение `HEAD^`, оно означает «обратиться к предку последнего коммита». Подробней описан синтаксис такой относительной адресации будет ниже, в разделе «Хэши, тэги, относительная адресация». Соответственно, `HEAD ---` ссылка на последний коммит. Ссылка `ORIG_HEAD` после «мягкого» резета указывает на оригинальный коммит.

Естественно, можно вернуться и на большую глубину коммитов,

«Жесткий» резет (ключ `--hard`) --- команда, которую следует использовать с осторожностью. `git reset --hard` вернет дерево проекта и индекс в состояние, соответствующее указанному коммиту, удалив изменения последующих коммитов:

```
git add .
git commit -m «destined to death»
git reset --hard HEAD~1 — больше никто и никогда не увидит этот позорный
коммит...
git reset --hard HEAD~3 — ...вернее, три последних коммита. Никто.
Никогда!
```

Если команда достигнет точки ветвления, удаления коммита не произойдет.

Для команд слияния или выкачивания последних изменений с удаленного репозитория примеры резета будут приведены в соответствующих разделах.

git revert --- отмена изменений, произведенных в прошлом отдельным коммитом

Возможна ситуация, в которой требуется отменить изменения, внесенные отдельным коммитом. `git revert` создает новый коммит, накладывающий обратные изменения.

Отменяем коммит, помеченный тегом:

```
git revert config-modify-tag
```

Отменяем коммит, используя его хэш:

```
git revert cgsjd2h
```

Для использования команды необходимо, чтобы состояние проекта не отличалось от состояния, зафиксированного последним коммитом.

git log --- разнообразная информация о коммитах в целом

Иногда требуется получить информацию об истории коммитов; коммитах, изменивших отдельный файл; коммитах за определенный отрезок времени и так далее. Для этих целей используется команда `git log`.

Простейший пример использования, в котором приводится короткая справка по всем коммитам, коснувшимся активной в настоящий момент ветки (о ветках и ветвлении подробно узнать можно ниже, в разделе «Ветвления и слияния»):

```
git log
```

Получить подробную информацию о каждом в виде патчей по файлам из коммитов можно, добавив ключ `-p` (или `-u`):

```
git log -p
```

Статистика изменения файлов, вроде числа измененных файлов, внесенных в них строк, удаленных файлов вызывается ключом `--stat`:

```
git log --stat
```

За информацию по созданиям, переименованиям и правам доступа файлов отвечает ключ `--summary`:

```
git log --summary
```

Чтобы просмотреть историю отдельного файла, достаточно указать в виде параметра его имя (кстати, в моей старой версии `git` этот способ не срабатывает, обязательно добавлять " --- " перед «README»):

```
git log README
```

или, если версия `git` не совсем свежая:

git log – README

Далее будет приводится только более современный вариант синтаксиса. Возможно указывать время, начиная в определенного момента («weeks», «days», «hours», «s» и так далее):

```
git log --since=«1 day 2 hours» README  
git log --since=«2 hours» README
```

изменения, касающиеся отдельной папки:

```
git log --since=«2 hours» dir/
```

Можно отталкиваться от тегов.

Все коммиты, начиная с тега v1:

```
git log v1...
```

Все коммиты, включающие изменения файла README, начиная с тега v1:

```
git log v1... README
```

Все коммиты, включающие изменения файла README, начиная с тега v1 и заканчивая тегом v2:

```
git log v1..v2 README
```

Интересные возможности по формату вывода команды предоставляет ключ `--pretty`.

Вывести на каждый из коммитов по строчке, состоящей из хэша (здесь --- уникального идентификатора каждого коммита, подробней --- дальше):

```
git log --pretty=oneline
```

Лаконичная информация о коммитах, приводятся только автор и комментарий:

```
git log --pretty=short
```

Более полная информация о коммитах, с именем автора, комментарием, датой создания и внесения коммита:

```
git log --pretty=full/fuller
```

В принципе, формат вывода можно определить самостоятельно:

```
git log --pretty=format:'FORMAT'
```

Определение формата можно поискать в разделе по `git log` из *Git Community Book* или справке. Красивый ASCII-граф коммитов выводится с использованием ключа `--graph`.

`git diff` --- отличия между деревьями проекта, коммитами и т.д.

Своего рода подмножеством команды `git log` можно считать команду `git diff`, определяющую изменения между объектами в проекте - деревьями (файлов и директорий).

Показать изменения, не внесенные в индекс:

```
git diff
```

Изменения, внесенные в индекс:

```
git diff --cached
```

Изменения в проекте по сравнению с последним коммитом:

```
git diff HEAD
```

Предпоследним коммитом:

```
git diff HEAD^
```

Можно сравнивать «головы» веток:

```
git diff master..experimental
```

или активную ветку с какой-либо:

```
git diff experimental
```

git show --- показать изменения, внесенные отдельным коммитом

Посмотреть изменения, внесенные любым коммитом в истории, можно командой `git show`:

```
git show COMMIT_TAG
```

git blame и git annotate --- команды, помогающие отслеживать изменения файлов

При работе в команде часто требуется выяснить, кто именно написал конкретный код. Удобно использовать команду `git blame`, выводящую построчную информацию о последнем коммите, коснувшемся строки, имя автора и хэш коммита:

```
git blame README
```

Можно указать и конкретные строки для отображения:

```
git blame -L 2,+3 README — выведет информацию по трем строкам, начиная со второй.
```

Аналогично работает команда `git annotate`, выводящая и строки, и информацию о коммитах, их коснувшихся:

```
git annotate README
```

git grep --- поиск слов по проекту, состоянию проекта в прошлом

`git grep`, в целом, просто дублирует функционал знаменитой юниксовой команды. Однако он позволяет слова и их сочетания искать в прошлом проекта, что бывает очень полезно.

Поиск слова `tst` в проекте:

```
git grep tst
```

Подсчитать число упоминаний `tst` в проекте:

```
git grep -c tst
```

Поиск в старой версии проекта:

```
git grep tst v1
```

Команда позволяет использовать логическое И и ИЛИ.

Найти строки, где упоминаются и первое слово, и второе:

```
git grep -e 'first' --and -e 'another'
```

Найти строки, где встречается хотя бы одно из слов:

```
git grep --all-match -e 'first' -e 'second'
```

Ветвление

git branch --- создание, перечисление и удаление веток

Работа с ветками --- очень легкая процедура в git, все необходимые механизмы сконцентрированы в одной команде:

Просто перечислить существующие ветки, отметив активную:

```
git branch
```

Создать новую ветку new-branch:

```
git branch new-branch
```

Удалить ветку, если та была залита (merged) с разрешением возможных конфликтов в текущую:

```
git branch -d new-branch
```

Удалить ветку в любом случае:

```
git branch -D new-branch
```

Переименовать ветку:

```
git branch -m new-name-branch
```

Показать те ветки, среди предков которых есть определенный коммит:

```
git branch --contains v1.2
```

git checkout --- переключение между ветками, извлечение файлов

Команда `git checkout` позволяет переключаться между последними коммитами (если упрощенно) веток:

```
checkout some-other-branch
```

Создаст ветку, в которую и произойдет переключение

```
checkout -b some-other-new-branch
```

Если в текущей ветке были какие-то изменения по сравнению с последним коммитом в ветке(HEAD), то команда откажется производить переключение, дабы не потерять произведенную работу.

Проигнорировать этот факт позволяет ключ `-f`:

```
checkout -f some-other-branch
```

В случае, когда изменения надо все же сохранить, следует использовать ключ `-m`. Тогда команда перед переключением попробует залить изменения в текущую ветку и, после разрешения возможных конфликтов, переключиться в новую:

```
checkout -m some-other-branch
```

Вернуть файл (или просто вытащить из прошлого коммита) позволяет команда вида:

Вернуть *somefile* к состоянию последнего коммита:

```
git checkout somefile
```

Вернуть *somefile* к состоянию на два коммита назад по ветке:

```
git checkout HEAD~2 somefile
```

git merge --- слияние веток (разрешение возможных конфликтов)

Слияние веток, в отличие от обычной практики централизованных систем, в git происходит практически каждый день. Естественно, что имеется удобный интерфейс к популярной операции.

Попробовать объединить текущую ветку и ветку new-feature:

```
git merge new-feature
```

В случае возникновения конфликтов коммита не происходит, а по проблемным файлам расставляются специальные метки а-ля svn; сами же файлы отмечаются в индексе как «не соединенные» (unmerged). До тех пор пока проблемы не будут решены, коммит совершить будет нельзя.

Например, конфликт возник в файле TROUBLE, что можно увидеть в `git status`.

Произошла неудачная попытка слияния:

```
git merge experiment
```

Смотрим на проблемные места:

```
git status
```

Разрешаем проблемы:

```
edit TROUBLE
```

Индексируем наши изменения, тем самым снимая метки:

```
git add .
```

Совершаем коммит слияния:

```
git commit
```

Вот и все, ничего сложного. Если в процессе разрешения вы передумали разрешать конфликт, достаточно набрать (это вернёт обе ветки в исходные состояния):

```
git reset --hard HEAD
```

Если же коммит слияния был совершен, используем команду:

```
git reset --hard ORIG_HEAD
```

git rebase --- построение ровной линии коммитов

Предположим, разработчик завел дополнительную ветку для разработки отдельной возможности и совершил в ней несколько коммитов. Одновременно по какой-либо причине в основной ветке также были совершены коммиты: например, в нее были залиты изменения с удаленного сервера, либо сам разработчик совершал в ней коммиты.

В принципе, можно обойтись обычным `git merge`. Но тогда усложняется сама линия разработки, что бывает нежелательно в слишком больших проектах, где участвует множество разработчиков.

Предположим, имеется две ветки, `master` и `topic`, в каждой из которых было совершено несколько коммитов начиная с момента ветвления. Команда `git rebase` берет коммиты из ветки `topic` и накладывает их на последний коммит ветки `master`.

Вариант, в котором явно указывается, что и куда накладывается:

```
git-rebase master topic
```

на master накладывается активная в настоящий момент ветка:

```
git-rebase master
```

После использования команды история становится линейной. При возникновении конфликтов при поочередном накладывании коммитов работа команды будет останавливаться, а в проблемные местах файлов появятся соответствующие метки. После редактирования --- разрешения конфликтов --- файлы следует внести в индекс командой `git add` и продолжить наложение следующих коммитов командой `git rebase --continue`. Альтернативными выходами будут команды `git rebase --skip` (пропустить наложение коммита и перейти к следующему) или `git rebase --abort` (отмена работы команды и всех внесенных изменений).

С ключом `-i` (`--interactive`) команда будет работать в интерактивном режиме. Пользователю будет предоставлена возможность определить порядок внесения изменений, автоматически будет вызывать редактор для разрешения конфликтов и так далее.

git cherry-pick --- применение к дереву проекта изменений, внесенных отдельным коммитом

Если ведется сложная история разработки, с несколькими длинными ветками разработками, может возникнуть необходимость в применении изменений, внесенных отдельным коммитом одной ветки, к дереву другой (активной в настоящий момент).

Изменения, внесенные указанным коммитом будут применены к дереву, автоматически проиндексированы и станут коммитом в активной ветке:

```
git cherry-pick BUG_FIX_TAG
```

Ключ `-n` показывает, что изменения надо просто применить к дереву проекта без индексации и создания коммита

```
git cherry-pick BUG_FIX_TAG -n
```

Прочие команды и необходимые возможности

Хэш --- уникальная идентификация объектов

В git для идентификации любых объектов используется уникальный (то есть с огромной вероятностью уникальный) хэш из 40 символов, который определяется хэширующей функцией на основе содержимого объекта. Объекты --- это все: коммиты, файлы, тэги, деревья. Поскольку хэш уникален для содержимого, например, файла, то и сравнивать такие файлы очень легко --- достаточно просто сравнить две строки в сорок символов.

Больше всего нас интересует тот факт, что хэши идентифицируют коммиты. В этом смысле хэш --- продвинутый аналог ревизий Subversion. Несколько примеров использования хэшей в качестве способа адресации:

найти разницу текущего состояния проекта и коммита за номером... сами видите, каким:

```
git diff f292ef5d2b2f6312bc45ae49c2dc14588eef8da2
```

То же самое, но оставляем только шесть первых символов. Git поймет, о каком коммите идет речь, если не существует другого коммита с таким началом хэша:

```
git diff f292ef5
```

Иногда хватает и четырех символов:

```
git diff f292
```

Читаем лог с коммита по коммит:

```
git log febc32...f292
```

Разумеется, человеку пользоваться хэшами не так удобно, как машине, именно поэтому были введены другие объекты --- тэги.

git tag --- тэги как способ пометить уникальный коммит

Тэг (tag) --- это объект, связанный с коммитом; хранящий ссылку на сам коммит, имя автора, собственное имя и некоторый комментарий. Кроме того, разработчик может оставлять на таких тегах собственную цифровую подпись.

Кроме этого в git представленные так называемые «легковесные тэги» (*lightweight tags*), состоящие только из имени и ссылки на коммит. Такие тэги, как правило, используются для упрощения навигации по дереву истории; создать их очень легко.

Создать «легковесный» тэг, связанный с последним коммитом; если тэг уже есть, то еще один создан не будет:

```
git tag stable-1
```

Пометить определенный коммит:

```
git tag stable-2 f292ef5
```

Удалить тег:

```
git tag -d stable-2
```

Перечислить тэги:

```
git tag -l
```

Создать тэг для последнего коммита, заменить существующий, если таковой уже был:

```
git tag -f stable-1.1
```

После создания тэга его имя можно использовать вместо хэша в любых командах вроде git diff, git log и так далее:

```
git diff stable-1.1...stable-1
```

Обычные тэги имеет смысл использовать для приложения к коммиту какой-либо информации, вроде номера версии и комментария к нему. Иными словами, если в комментарии к коммиту пишешь «исправил такой-то баг», то в комментарии к тэгу по имени «v1.0» будет что-то вроде «стабильная версия, готовая к использованию».

Создать обычный тэг для последнего коммита; будет вызван текстовый редактор для составления комментария:

```
git tag -a stable
```

Создать обычный тэг, сразу указав в качестве аргумента комментарий:

```
git tag -a stable -m "production version"
```

Команды перечисления, удаления, перезаписи для обычных тэгов не отличаются от команд для «легковесных» тэгов.

Относительная адресация

Вместо ревизий и тэгов в качестве имени коммита можно опираться на еще один механизм --- относительную адресацию. Например, можно обратиться прямо к предку последнего коммита ветки master:

```
git diff master^
```

Если после «птички» поставить цифру, то можно адресоваться по нескольким предкам коммитов слияния:

найти изменения по сравнению со вторым предком последнего коммита в master; HEAD здесь --- указатель на последний коммит активной ветки:

```
git diff HEAD^2
```

Аналогично, тильдой можно просто указывать, насколько глубоко в историю ветки нужно погрузиться: что привнес «дедушка» нынешнего коммита:

```
git diff master^^
```

То же самое:

```
git diff master~2
```

Обозначения можно объединять, чтобы добраться до нужного коммита:

```
git diff master~3^~2
```

```
git diff master~6
```

файл .gitignore --- объясняем git, какие файлы следует игнорировать

Иногда по директориям проекта встречаются файлы, которые не хочется постоянно видеть в сводке `git status`. Например, вспомогательные файлы текстовых редакторов, временные файлы и прочий мусор.

Заставить `git status` игнорировать определенные файлы можно, создав в корне или глубже по дереву (если ограничения должны быть только в определенных директориях) файл `.gitignore`. В этих файлах можно описывать шаблоны игнорируемых файлов определенного формата.

Пример содержимого такого файла:

```
#комментарий к файлу .gitignore
#игнорируем сам .gitignore
.gitignore
#все html-файлы...
*.html
#... кроме определенного
!special.html
#не нужны объектники и архивы
*. [ao]
```

Существуют и другие способы указания игнорируемых файлов, о которых можно узнать из справки `git help gitignore`.

Серверные команды репозитория

; `git update-server-info` : Команда создает вспомогательные файлы для dumb-сервера в `$GIT_DIR/info` и `$GIT_OBJECT_DIRECTORY/info` каталогах, чтобы помочь клиентам узнать, какие ссылки и пакеты есть на сервере.

; git count-objects : Проверка, сколько объектов будет потеряно и объём освобождаемого места при перепаковке репозитория.

; git gc : Переупаковка локального репозитория.

Рецепты

Создание пустого репозитория на сервере

```
repo="repo.git"
mkdir $repo
cd $repo
git init --bare
chown git. -R ./
cd ../
```

Импорт svn репозитория на Git-сервер

```
repo="repo.svn"
svnserver="http://svn.calculate.ru"
git svn clone -s $svnserver/$repo $repo
mv $repo/.git/refs/remotes/tags $repo/.git/refs/tags
rm -rf $repo/.git/refs/remotes
rm -rf $repo/.git/svn
mv $repo/.git $repo.git
rm -rf $repo
cd $repo.git
chown git. -R ./
cd ../
```

Ссылки

- [правка коммитов](#)
- [Перенос SVN-репозитория в git](#)
- [Учебник-введение в git](#)
- [Руководство пользователя GIT](#)
- [20 повседневных команд git](#)
- [Внутренности git](#)
- [Введение в структуру хранилища git](#)
- [Практическое введение в git](#)
- [Работа с git для начинающих](#)
- [Переходим с SVN на Git](#)
- [XX полезных советов для пользователей Git среднего уровня. Часть 1](#)
- [XX полезных советов для пользователей Git среднего уровня. Часть 2](#)
- [Внешние зависимости в гите: submodule или subtree?](#)
- [Командная работа в Git](#)

Руководство по оформлению программ на Python

Author: Guido van Rossum

Оригинальная статья на английском: <http://www.python.org/doc/essays/styleguide.html> Python Style Guide
Перевод выполнен компанией «Калкулэйт».

Общие замечания

Это руководство - о логике программирования. Важно следовать этому руководству в стиле программы. Ещё более важна логичность всего проекта. Логичность в пределах одного модуля или функции - важнейшее требование. Но важнее всего знать, когда отступить от стиля и логичности. Иногда это руководство просто неприменимо. Если вы затрудняетесь, посмотрите на примеры и решите, что лучше.

Разметка

Отступы

Используйте параметры по умолчанию: 4 пробела на один отступ. Для очень старого кода, в который вы не желаете сильно вмешиваться, можете продолжать использовать отступы в 8 символов.

Табуляция или пробелы?

Никогда не смешивайте пробелы с табуляцией. Самый популярный способ отступов в Питоне - использовать только пробелы. Второй самый популярный способ - только табуляторы. Код, в котором отступы пробелами и табуляторами перемешаны, надо перевести к отступам пробелами. Когда вы запускаете Питон с параметром `-t`, то при использовании смешанных отступов выдаются предупреждения; если задать опцию `-tt`, то предупреждения станут ошибками. Использование этих опций очень рекомендуется!

Максимальная длина строки

До сих пор существует много устройств, длина строки в которых ограничена 80-ю символами. Поэтому, пожалуйста, установите максимальную длину для всех строк 79 символов.

Наиболее предпочтительный путь для переноса длинных строк - использование встроенного в Питон продолжения строк внутри скобок. Если это необходимо, вы можете использовать дополнительную пару скобок вокруг выражения, но иногда обратный слэш выглядит лучше. Убедитесь, что для перенесенных строк установлен правильный отступ:

```
class Rectangle(Blob):
    def __init__(self, width, height,
                 color='black', emphasis=None, highlight=0):
        if width == 0 and height == 0 and \
           color == 'red' and emphasis == 'strong' or \
           highlight > 100:
            raise ValueError, "sorry, you lose"
        if width == 0 and height == 0 and (color == 'red' or
                                         emphasis is None):
            raise ValueError, "I don't think so"
        Blob.__init__(self, width, height,
                      color, emphasis, highlight)
```

Пустые строки

Отделяйте функции верхнего уровня и объявления классов двумя пустыми строками. Определения методов внутри класса отделяются одной пустой строкой. Дополнительно пустые строки можно использовать для отделения групп родственных функций. Пустые строки можно опускать внутри связки из односторонних определений (набора абстрактных методов).

Если пустые строки используются для отделения методов в классе, то вставляйте также пустую строку между строкой `"class..."` и определением первого метода.

Используйте пустые строки в функциях для указания логических блоков, но не переусердствуйте.

Пробелы в выражениях и операторах

Я (Pet Peeves) **ненавижу** пробелы в следующих местах:

- Сразу после скобок:

```
spam( ham[ 1 ], { eggs: 2 } ).
```

Всегда пишите так:

```
spam(ham[1], {eggs: 2}).
```

- Непосредственно перед двоеточием, точкой с запятой, запятой:

```
if x == 4 : print x , y ; x , y = y , x.
```

Всегда пишите:

```
if x == 4: print x, y; x, y = y, x.
```

- Перед открывающей скобкой при вызове функции:

```
spam (1)
```

Всегда пишите:

```
spam(1)
```

- Перед открывающими скобками индекса или разреза:

```
dict ['key'] = list [index].
```

Всегда пишите:

```
dict['key'] = list[index].
```

- Более чем один пробел вокруг присваивания или другого оператора:

```
x           = 1
y           = 2
long_variable = 3
```

Всегда пишите так:

```
x = 1
y = 2
long_variable = 3
```

Другие рекомендации

ВСЕГДА окружайте эти бинарные операторы одинарными пробелами:

- присваивание
- сравнения (==,
- Булевы (and, or, not).

Используйте наилучший по вашему мнению выбор при вставке пробелов вокруг арифметических операторов. Всегда будьте последовательны при вставке пробелов в обоих частях бинарного оператора:

```
i = i+1
submitted = submitted + 1
x = x*2 - 1
hypot2 = x*x + y*y
c = (a+b) * (a-b)
c = (a + b) * (a - b)
```

Не используйте пробелы вокруг знака '=' в случае указания значения по умолчанию:

```
def complex(real, imag=0.0):
    return magic(r=real, i=imag)
```

Комментарии

Комментарии, которые противоречат коду, хуже, чем код без комментариев вообще.

Всегда поддерживайте актуальность комментариев, изменяйте их каждый раз при изменениях кода!

Если комментарий - фраза или предложение, его первая буква должна быть заглавной, если только это не идентификатор.

Если это короткий комментарий, то точку в конце лучше опустить. Блоковые комментарии обычно состоят из нескольких параграфов, состоящих из полных предложений, а каждое предложение должно заканчиваться точкой.

Используйте два пробела после точки в конце предложения.

Если вы на 120% не уверены, что ваш код никогда не будут читать люди, не говорящие на вашем языке, - пишите по-английски.

Блоковые комментарии

Блоковые комментарии обычно распространяются на код, который следует сразу за ними и располагается на одном с ними уровне отступа. Каждая строка в блоке комментария начинается с # и одного пробела. Параграфы в комментариях отделяются строкой с одним символом #.

Лучше всего отделять блоковые комментарии пустыми строками сверху и снизу. Если это блок комментария, который начинает новый раздел функций, то сверху можно поставить две пустых строки.

Внутристроковые комментарии

Внутристроковые комментарии - комментарии на той же строке, что и оператор. Внутристроковые комментарии надо использовать умеренно. Внутристроковые комментарии надо отделять как минимум двумя пробелами от оператора, они начинаются с # и одного пробела.

Внутристроковые комментарии излишни и отвлекают, если их значение и так очевидно:

```
x = x+1 # Увеличение x на единицу
```

Но иногда может быть полезно:

```
x = x+1 # Для компенсации толщины рамки
```

Строки документации

Все модули, как правило, имеют строки документации. Все функции и классы, экспортруемые из модуля, также должны иметь строки документации.

Публичные методы (включая конструктор *init*) тоже должны иметь строки документации.

Строки документации должны быть написаны в стиле "Usage" - информации, которая обычно выдается программами на экран при вызове с ключом -h или -help.

Всегда используйте """тройные сдвоенные кавычки""" для выделения строк документации.

Есть два вида строк документации - однострочные и многостроковые.

Однострочные

```
def find_root():
    """Решить уравнение и вернуть его корень."""
    ...
```

- Используйте """тройные сдвоенные кавычки"""
- закрывающие кавычки на той же строке

- никаких пустых строк до или после комментария
- Фраза заканчивается точкой.
- Пишите в стиле команды, приказа ("Сделать" это, "вернуть" то-то).
- Никогда не пишите что-то типа "Функция возвращает..."

Многостроковые

Начинаются одной обобщающей строкой, за которой следует пустая строка и более полное описание.

Рекомендуется перед закрывающими тройными кавычками вставлять пустую строку.

Каждый аргумент лучше начинать с новой строки и отделять от описания двумя тире:

```
def complex(real=0.0, imag=0.0):
    """Form a complex number.

    Keyword arguments:
    real -- the real part (default 0.0)
    imag -- the imaginary part (default 0.0)

    """
    if imag == 0.0 and real == 0.0: return complex_zero
    ...
```

Поддержка контроля версий

Если вы используете RCS или CVS, то пишите следующим образом:

```
__version__ = "$Revision: 6104 $"
# $Source$
```

Вставляйте это после строк документации перед началом кода, отделяя сверху и снизу пустой строкой.

Именование

Способы именования в библиотеках Питона - это всегда маленький хаос. Здесь никогда не будет полной логичности и порядка. Тем не менее дадим несколько рекомендаций.

Стили именования

Стилей много. Наиболее распространены следующие:

- x (одна маленькая буква)
- X (одна большая буква)
- lowercase - маленькими буквами
- lower_case_with_underscores - маленькими буквами с подчеркиваниями
- UPPERCASE - большими буквами
- UPPER_CASE_WITH_UNDERSCORES - большими буквами с подчеркиваниями
- CapitalizedWords (or CapWords) - Заглавные буквы слов
- mixedCase - смешанный
- Capitalized_Words_With_Underscores - Заглавные буквы слов + подчеркивания (какое убожество!) В дополнение существуют следующие специальные формы с символами подчеркивания:
 - _single_leading_underscore: внутреннее использование, ("from M import *" не будет импортировать такие имена)
 - single_trailing_underscore_: исп. для предотвращения конфликтов с зарезервированными словами Питона (Tkinter.Toplevel(master, class_="ClassName")).
 - __double_leading_underscore: private-имена класса в Python 1.4.
 - __double_leading_and_trailing_underscore__: "волшебные" объекты, например *init*, *import* или *file*.

Стили предписаний

Названия модулей

Имена модулей можно писать в стиле "MixedCase" или "lowercase".

Модули, которые экспортят один класс, обычно называют в стиле MixedCase, а имя модуля совпадает с именем класса (например, стандартный модуль StringIO).

Модули, которые экспортят множество функций, обычно называют в стиле lowercase.

В случае когда модуль расширения написанный на С или С++ имеет сопровождающий его модуль на Питоне, который представляет собой интерфейс высокого уровня (объектно ориентированный), питоновский модуль называют в виде "ModuleName", а модуль С/С++ - "_modulename".

Class Names

Имена классов обычно используют стиль "CapWords". Классы для внутреннего пользования начинаются с подчеркивания.

Exception Names

Если модуль вызывает одно исключение при любой ошибке, то его обычно называют "error" или "Error". Например встроенные модули расширений используют "error" (os.error), а питоновские модули - "Error" (xdrlib.Error).

Function Names

Обычные функции, которые экспортят модуль, могут быть в любом стиле "CapWords" или "lowercase" (или "lower_case_with_underscores"). Стиль "CapWords" используется для функций, которые предоставляют наибольшую функциональность. (nstools.WorldOpen()), а "lowercase" - для мелких функций (pathhack.kos_root()).

Global Variable Names

То же, что и для экспортруемых функций.

Method Names

Как для всех функций.

Программный Raid

Руководство по созданию программного Raid5 массива.

Создание raid5 массива

```
mdadm --create /dev/md0 --level=5 --raid-devices=3 /dev/sdb1 /dev/sdd1  
/dev/sde1
```

Добавление устройства

```
mdadm /dev/md0 -a /dev/sdc1
```

Подключение добавленного устройства к raid5 массиву

```
mdadm -G /dev/md0 -n4
```

Просмотр состояния raid5 массива

```
mdadm -Q --detail /dev/md0
```

Форматирование raid5 массива (XFS)

```
mkfs.xfs /dev/md0
```

Монтирование raid5 массива

```
mount /dev/md0 /mnt/floppy
```

Расширение файловой системы XFS при увеличении размера массива

```
xfs_growfs /dev/md0
```

Остановка raid5 массива

```
mdadm -S /dev/md0
```

Поиск raid5 дисков и их подключение после остановки

```
mdadm --assemble --scan
```

Включение raid5 массива при загрузке системы

Генерация mdadm.conf

Должно существовать устройство /dev/md0.

```
mdadm --detail --scan > /etc/mdadm.conf
```

Включения модуля в автозагрузку

Добавляем в /etc/conf.d/modules:

```
modules="raid5"
```

Автозапуск mdraid

```
rc-update add mdraid boot
```

Руководство по Iptables

- [Руководство по Iptables](#)
- [Глава 1. Введение](#)
- [1.1. Почему было написано данное руководство](#)
- [1.2. Как он был написан](#)
- [1.3. Термины, используемые в данном документе](#)
- [Глава 2. Подготовка](#)
- [2.1. Где взять iptables](#)
- [2.2. Настройка ядра](#)
- [2.3. Установка пакета](#)
- [Глава 3. Порядок прохождения таблиц и цепочек.](#)
- [3.1. Общие положения](#)
- [3.2. Таблица Mangle](#)
- [3.3. Таблица Nat](#)
- [3.4. Таблица Filter](#)
- [Глава 4. Механизм определения состояний](#)

- [4.1. Введение](#)
- [4.2. Таблица трассировщика](#)
- [4.3. Состояния в пространстве пользователя](#)
- [4.4. TCP соединения](#)
- [4.5. UDP соединения](#)
- [4.6. ICMP соединения](#)
- [4.7. Поведение по-умолчанию](#)
- [4.8. Трассировка комплексных протоколов](#)
- [Глава 5. Сохранение и восстановление больших наборов правил](#)
- [5.1. Плюсы](#)
- [5.2. И минусы](#)
- [5.3. iptables-save](#)
- [5.4. iptables-restore](#)
- [Глава 6. Как строить правила](#)
- [6.1. Основы](#)
- [6.2. Таблицы](#)
- [6.3. Команды](#)
- [6.4. Критерии](#)
- [6.5. Действия и переходы](#)
- [Глава 7. Файл rc.firewall](#)
- [7.1. Пример rc.firewall](#)
- [7.2. Описание сценария rc.firewall](#)
- [Глава 8. Примеры сценариев](#)
- [8.1. Структура файла rc.firewall.txt](#)
- [8.1.1. Структура](#)
- [8.2. rc.firewall.txt](#)
- [8.3. rc.DMZ.firewall.txt](#)
- [8.4. rc.DHCP.firewall.txt](#)
- [8.5. rc.UTIN.firewall.txt](#)
- [8.6. rc.test-iptables.txt](#)
- [8.7. rc.flush-iptables.txt](#)
- [8.8. Limit-match.txt](#)
- [8.9. Pid-owner.txt](#)
- [8.10. Sid-owner.txt](#)
- [8.11. Ttl-inc.txt](#)
- [8.12. Iptables-save ruleset](#)
- [Приложение А. Детальное описание специальных команд](#)
- [A.1. Вывод списка правил](#)
- [A.2. Изменение и очистка ваших таблиц](#)
- [Приложение В. Общие проблемы и вопросы](#)
- [B.1. Проблемы загрузки модулей](#)
- [B.2. Пакеты со статусом NEW и соброшенным битом SYN](#)
- [B.3. SYN/ACK - пакеты и пакеты со статусом NEW](#)
- [B.4. Поставщики услуг Internet, использующие зарезервированные IP-адреса](#)
- [B.5. Как разрешить прохождение DHCP запросов через iptables](#)
- [B.6. Проблемы mIRC DCC](#)
- [Приложение С. Типы ICMP](#)
- [Приложение D. Ссылки на другие ресурсы](#)
- [Приложение Е. Благодарности](#)

Глава 1. Введение

1.1. Почему было написано данное руководство

Скажем так, я посчитал, что существует досадный пробел в HOWTO по части информации об iptables и функциях сетевого фильтра (netfilter), реализованных в новой серии ядер 2.4.x Linux. Кроме всего прочего, я попытался ответить на некоторые вопросы по поводу новых возможностей, например проверки состояния пакетов (state matching). Большинство из них проиллюстрированы в файле скрипта [rc.firewall.txt](#), который вы можете вставить в `/etc/rc.d/`. Для тех, кому интересно, готов сообщить, что этот файл первоначально был основан на masquerading HOWTO.

Там же вы найдете небольшой сценарий [rc.flush-iptables.txt](#), написанный мною, который вы можете использовать, для своих нужд, при необходимости расширяя под свою конфигурацию.

1.2. Как он был написан

Я консультировался с Марком Бучером (Marc Boucher) и другими членами команды разработчиков netfilter. Пользуясь случаем, выражая огромную признательность за их помощь в создании данного руководства, которое изначально было написано для boingworld.com, а теперь доступно на моем персональном сайте frozentux.net. С помощью этого документа вы пройдете процесс настройки шаг за шагом и, надеюсь, что к концу изучения его вы будете знать о пакете iptables значительно больше. Большая часть материала базируется на файле `rc.firewall.txt`, так как я считаю, что рассмотрение примера -- лучший способ изучения iptables. Я пройду по основным цепочкам правил в порядке их следования. Это несколько усложняет изучение, зато изложение становится логичнее. И, всякий раз, когда у вас возникнут затруднения, вы можете обращаться к этому руководству.

1.3. Термины, используемые в данном документе

Этот документ содержит несколько терминов, которые следует пояснить прежде, чем вы столкнетесь с ними.

DNAT - от англ. Destination Network Address Translation - Изменение Сетевого Адреса Получателя. DNAT - это изменение адреса назначения в заголовке пакета. Зачастую используется в паре с SNAT. Основное применение -- использование единственного реального IP-адреса несколькими компьютерами для выхода в Интернет и предоставления дополнительных сетевых услуг внешним клиентам.

"Поток" (Stream) - под этим термином подразумевается соединение, через которое передаются и принимаются пакеты. Я использовал этот термин для обозначения соединений, через которые передается по меньшей мере 2 пакета в обеих направлениях. В случае TCP это может означать соединение, через которое передается SYN пакет и затем принимается SYN/ACK пакет. Но это так же может подразумевать и передачу SYN пакета и прием сообщения ICMP Host unreachable. Другими словами, я использую этот термин в достаточно широком диапазоне применений.

SNAT - от англ. Source Network Address Translation - Изменение Сетевого Адреса Отправителя. SNAT - это изменение исходного адреса в заголовке пакета. Основное применение -- использование единственного реального IP-адреса несколькими компьютерами для выхода в Интернет. В настоящее время диапазон реальных IP-адресов, по стандарту IPv4, недостаточно широк, и его не хватает на всех (переход на IPv6 разрешит эту проблему).

"Состояние" (State) - под этим термином подразумевается состояние, в котором находится пакет, согласно [RFC 793](#), а также трактовкам, используемым в netfilter/iptables. Хочу обратить ваше внимание на тот факт, что определения состояний пакетов, как для внутренних так и для внешних состояний, используемые Netfilter, не полностью соответствуют указанному выше RFC 793.

"Пространство пользователя" (User space) - под этим термином я подразумеваю все, что расположено за пределами ядра, например: команда `iptables -h` выполняется за пределами ядра, в то время как команда `iptables -A FORWARD -p tcp -j ACCEPT` выполняется (частично) в пространстве ядра, поскольку она добавляет новое правило к имеющемуся набору.

"Пространство ядра" (Kernel space) - в большей или меньшей степени является утверждением, обратным термину "Пространство пользователя". Подразумевает место исполнения - в пределах ядра.

"Userland" - см. *Пространство пользователя*.

Глава 2. Подготовка

Целью данной главы является оказание помощи в понимании той роли, которую netfilter и **iptables** играют в Linux сегодня. Так же она должна помочь вам установить и настроить межсетевой экран (firewall).

2.1. Где взять iptables

Пакеты iptables могут быть загружены с домашней страницы [проекта Netfilter](#). Кроме того, для работы iptables соответствующим образом должно быть сконфигурировано ядро вашей Linux-системы. Настройка ядра будет обсуждаться ниже.

2.2. Настройка ядра

Для обеспечения базовых возможностей **iptables**, с помощью утилиты **make config** или ей подобных (**make menuconfig** или **make xconfig** прим. перев.), в ядро должны быть включены следующие опции: **CONFIG_PACKET** - Эта опция необходима для приложений, работающих непосредственно с сетевыми устройствами, например: tcpdump или snort.

Строго говоря, опция **CONFIG_PACKET** не требуется для работы iptables, но, поскольку она используется довольно часто, я включил ее в список. Если вам эта опция не нужна, то можете ее не включать.

- **CONFIG_NETFILTER** - Эта опция необходима, если вы собираетесь использовать компьютер в качестве сетевого экрана (firewall) или шлюза (gateway) в Интернет. Другими словами, вам она определенно понадобится, иначе зачем тогда читать это руководство!

И конечно, нужно добавить драйверы для ваших устройств, т.е. для карты Ethernet, PPP и SLIP. Эти опции необходимы для обеспечения базовых возможностей **iptables**, для получения дополнительных возможностей придется включить в ядро некоторые дополнительные опции. Ниже приводится список опций для ядра 2.4.9 и их краткое описание:

- **CONFIG_IP_NF_CONNTRACK** - Трассировка соединений. Трассировка соединений, среди всего прочего, используется при трансляции сетевых адресов и маскарадинге (NAT и Masquerading). Если вы собираетесь строить сетевой экран (firewall) для локальной сети, то вам определенно потребуется эта опция. К примеру, этот модуль необходим для работы [rc.firewall.txt](#).
- **CONFIG_IP_NF_FTP** - Трассировка FTP соединений. Обмен по FTP идет слишком интенсивно, чтобы использовать обычные методы трассировки. Если не добавить этот модуль, то вы столкнетесь с трудностями при передаче протокола FTP через сетевой экран (firewall).
- **CONFIG_IP_NF_IPTABLES** - Эта опция необходима для выполнения операций фильтрации, преобразования сетевых адресов (NAT) и маскарадинга (masquerading). Без нее вы вообще ничего не сможете делать с iptables.
- **CONFIG_IP_NF_MATCH_LIMIT** - Этот модуль необязателен, однако он используется в примерах [rc.firewall.txt](#). Он предоставляет возможность ограничения количества проверок для некоторого правила. Например, `-m limit --limit 3/minute` указывает, что заданное правило может пропустить не более 3-х пакетов в минуту. Таким образом, данный модуль может использоваться для защиты от нападений типа "Отказ в обслуживании".
- **CONFIG_IP_NF_MATCH_MAC** - Этот модуль позволит строить правила, основанные на MAC-адресации. Как известно, каждая сетевая карта имеет свой собственный уникальный Ethernet-адрес, таким образом, существует возможность блокировать пакеты, поступающие с определенных MAC-адресов (т.е. с определенных сетевых карт). Следует, однако, отметить что данный модуль не используется в [rc.firewall.txt](#) или где либо еще в данном руководстве.

- **CONFIG_IP_NF_MATCH_MARK** - Функция маркировки пакетов MARK. Например, при использовании функции MARK мы получаем возможность пометить требуемые пакеты, а затем, в других таблицах, в зависимости от значения метки, принимать решение о маршрутизации помеченного пакета. Более подробное описание функции MARK приводится ниже в данном документе.
- **CONFIG_IP_NF_MATCH_MULTIPORT** - Этот модуль позволит строить правила с проверкой на принадлежность пакета к диапазону номеров портов источника/приемника.
- **CONFIG_IP_NF_MATCH_TOS** - Этот модуль позволяет строить правила, отталкиваясь от состояния поля TOS в пакете. Поле TOS устанавливается для Type Of Service. Так же становится возможным устанавливать и сбрасывать биты этого поля в собственных правилах в таблице mangle или командами ip/tc.
- **CONFIG_IP_NF_MATCH_TCPMSS** - Эта опция добавляет возможность проверки поля MSS в TCP-пакетах.
- **CONFIG_IP_NF_MATCH_STATE** - Это одно из самых серьезных усовершенствований по сравнению с **ipchains**. Этот модуль предоставляет возможность управления TCP пакетами, основываясь на их состоянии (state). К примеру, допустим, что мы имеем установленное TCP соединение, с траффиком в оба конца, тогда пакет полученный по такому соединению будет считаться **ESTABLISHED** (установленное соединение -- прим. ред). Эта возможность широко используется в примере [rc.firewall.txt](#).
- **CONFIG_IP_NF_MATCH_UNCLEAN** - Этот модуль реализует возможность дополнительной проверки IP, TCP, UDP и ICMP пакетов на предмет наличия в них несоответствий, "странных", ошибок. Установив его мы, к примеру, получим возможность "отсекать" подобного рода пакеты. Однако хочется отметить, что данный модуль пока находится на экспериментальной стадии и не во всех случаях будет работать одинаково, поэтому никогда нельзя будет быть уверенным, что мы не "бросили" вполне правильные пакеты.
- **CONFIG_IP_NF_MATCH_OWNER** - Проверка "владельца" соединения (socket). Для примера, мы можем позволить только пользователю root выходить в Internet. Этот модуль был написан как пример работы с **iptables**. Следует заметить, что данный модуль имеет статус экспериментального и может не всегда выполнять свои функции.
- **CONFIG_IP_NF_FILTER** - Реализация таблицы filter в которой в основном и осуществляется фильтрация. В данной таблице находятся цепочки **INPUT**, **FORWARD** и **OUTPUT**. Этот модуль обязателен, если вы планируете осуществлять фильтрацию пакетов.
- **CONFIG_IP_NF_TARGET_REJECT** - Добавляется действие **REJECT**, которое производит передачу ICMP-сообщения об ошибке в ответ на входящий пакет, который отвергается заданным правилом. Запомните, что TCP соединения, в отличие от UDP и ICMP, всегда завершаются или отвергаются пакетом TCP RST.
- **CONFIG_IP_NF_TARGET_MIRROR** - Возможность отправки полученного пакета обратно (отражение). Например, если назначить действие MIRROR для пакетов, идущих в порт HTTP через нашу цепочку INPUT (т.е. на наш WEB-сервер прим. перев.), то пакет будет отправлен обратно (отражен) и, в результате, отправитель увидит свою собственную домашнюю страничку. (Тут одни сплошные "если": Если у отправителя стоит WEB-сервер, если он работает на том же порту, если у отправителя есть домашняя страничка, и т.д. . Суть-то собственно сводится к тому, что с точки зрения отправителя все выглядит так, как будто бы пакет он отправил на свою собственную машину, а проще говоря, действие MIRROR меняет местами адрес отправителя и получателя и выдает измененный пакет в сеть прим. перев.)
- **CONFIG_IP_NF_NAT** - Трансляция сетевых адресов в различных ее видах. С помощью этой опции вы сможете дать выход в Интернет всем компьютерам вашей локальной сети, имея лишь один уникальный IP-адрес. Эта опция необходима для работы примера [rc.firewall.txt](#).

- **CONFIG_IP_NF_TARGET_MASQUERADE** - Маскарадинг. В отличие от NAT, маскарадинг используется в тех случаях, когда заранее неизвестен наш IP-адрес в Интернете, т.е. для случаев DHCP, PPP, SLIP или какого-либо другого способа подключения, подразумевающего динамическое получение IP-адреса. Маскарадинг дает несколько более высокую нагрузку на компьютер, по сравнению с NAT, однако он работает в ситуациях, когда невозможно заранее указать собственный внешний IP-адрес.
- **CONFIG_IP_NF_TARGET_REDIRECT** - Перенаправление. Обычно это действие используется совместно с проксированием. Вместо того, чтобы просто пропустить пакет дальше, это действие перенаправляет пакет на другой порт сетевого экрана (прокси-серверу прим. перев.). Другими словами, таким способом мы можем выполнять "*прозрачное проксирование*".
- **CONFIG_IP_NF_TARGET_LOG** - Добавляет действие LOG в **iptables**. Мы можем использовать этот модуль для фиксации отдельных пакетов в системном журнале (syslog). Эта возможность может оказаться весьма полезной при отладке ваших сценариев.
- **CONFIG_IP_NF_TARGET_TCPCMSS** - Эта опция может использоваться для преодоления ограничений, накладываемых некоторыми провайдерами (*Internet Service Providers*), которые блокируют ICMP Fragmentation Needed пакеты. В результате таких ограничений серверы провайдеров могут не передавать web-страницы, ssh может работать, в то время как scp обрывается после установления соединения и пр. Для преодоления подобного рода ограничений мы можем использовать действие TCPCMSS ограничивая значение MSS (*Maximum Segment Size*) (обычно MSS ограничивается размером MTU исходящего интерфейса минус 40 байт прим. перев.). Таким образом мы получаем возможность преодолеть то, что авторы **netfilter** называют "преступной безмозглостью провайдеров или серверов" ("*criminally braindead ISPs or servers*") в справке по конфигурации ядра.
- **CONFIG_IP_NF_COMPAT_IPCHAINS** - Добавляет совместимость с более старой технологией ipchains. Вполне возможно, что подобного рода совместимость будет сохранена и в ядрах серии 2.6.x.
- **CONFIG_IP_NF_COMPAT_IPFWADM** - Добавляет совместимость с **ipfwadm**, не смотря на то что это очень старое средство построения брандмауэров.

Как вы можете видеть, я дал краткую характеристику каждому модулю. Данные опции доступны в ядре версии 2.4.9. Если вам потребуются дополнительные возможности - советую обратить внимание на расширения *patch-o-matic*, которые добавляют достаточно большое количество дополнительных функций к Netfilter. *Patch-o-matic* - это набор дополнений, которые, как предполагается, в будущем будут включены в состав ядра. Для работы сценария [rc.firewall.txt](#) вам необходимо будет добавить в ядро следующие опции или собрать соответствующие подгружаемые модули. За информацией по опциям, необходимым для работы других сценариев, обращайтесь к приложению с примерами этих сценариев.

- **CONFIG_PACKET**
- **CONFIG_NETFILTER**
- **CONFIG_IP_NF_CONNTRACK**
- **CONFIG_IP_NF_FTP**
- **CONFIG_IP_NF_IRC**
- **CONFIG_IP_NF_IPTABLES**
- **CONFIG_IP_NF_FILTER**
- **CONFIG_IP_NF_NAT**
- **CONFIG_IP_NF_MATCH_STATE**
- **CONFIG_IP_NF_TARGET_LOG**
- **CONFIG_IP_NF_MATCH_LIMIT**
- **CONFIG_IP_NF_TARGET_MASQUERADE**

Выше приведен список минимально необходимых опций ядра для сценария [rc.firewall.txt](#) Перечень опций, необходимых для других примеров сценариев вы сможете найти в соответствующих разделах ниже. Сейчас же мы остановимся на главном сценарии и начнем его изучение.

2.3. Установка пакета

В первую очередь посмотрим как собрать (скомпилировать) пакет **iptables**. Сборка пакета в значительной степени зависит от конфигурации ядра и вы должны это понимать. Некоторые дистрибутивы предполагают предустановку пакета **iptables**, один из них - Red Hat. Однако, в RedHat этот пакет по умолчанию выключен, поэтому ниже мы рассмотрим как его включить в данном и в других дистрибутивах.

2.3.1. Сборка пакета

Для начала пакет с исходными текстами **iptables** нужно распаковать. Мы будем рассматривать пакет iptables 1.2.6a и ядро серии 2.4. Распакуем как обычно, командой `bzip2 -cd iptables-1.2.6a.tar.bz2 | tar -xvf -` (распаковку можно выполнить также командой `tar -xjvf iptables-1.2.6a.tar.bz2`). Если распаковка прошла удачно, то пакет будет размещен в каталоге `iptables-1.2.6a/INSTALL`, который содержит подробную информацию по сборке и установке пакета.

Далее необходимо проверить включение в ядро дополнительных модулей и опций. Шаги, описываемые здесь, будут касаться только наложения "заплат" (patches) на ядро. На этом шаге мы установим обновления, которые, как ожидается, будут включены в ядро в будущем.

Некоторые из них находятся пока на экспериментальной стадии и наложение этих заплат может оказаться не всегда оправданной, однако среди них есть чрезвычайно интересные функции и действия.

Выполним этот шаг, набрав команду (естественно, обладая правами пользователя root):

```
make pending-patches KERNEL_DIR=/usr/src/linux/
```

Переменная `KERNEL_DIR` должна содержать путь к исходным текстам вашего ядра. Обычно это `/usr/src/linux/`. Если исходные тексты у вас расположены в другом месте, то, соответственно, вы должны указать свой путь.

Здесь предполагается выполнить несколько обновлений и дополнений, которые определенно войдут в состав ядра, но несколько позднее, сейчас же мы возьмем их отсюда выполнив команду:

```
make most-of-pom KERNEL_DIR=/usr/src/linux/
```

В процессе выполнения вышеприведенной команды у вас будет запрашиваться подтверждение на обновление каждого раздела из того, что в мире netfilter называется **patch-o-matic**. Чтобы установить все "заплатки" из **patch-o-matic**, вам нужно выполнить следующую команду:

```
make patch-o-matic KERNEL_DIR=/usr/src/linux/
```

Не забудьте внимательно и до конца прочитать справку по каждой "заплатке" до того как вы будете устанавливать что-либо, поскольку одни "заплатки" могут оказаться несовместимы с другими, а некоторые - при совместном наложении даже разрушить ядро.

Вы можете вообще пропустить обновление ядра, другими словами особой нужды в таком обновлении нет, однако **patch-o-matic** содержит действительно интересные обновления, и у вас вполне может возникнуть желание посмотреть на них. Ничего страшного не случится, если вы запустите эти команды и посмотрите какие обновления имеются.

После завершения обновления, вам необходимо будет пересобрать ядро, добавив в него только что установленные обновления. Не забудьте сначала выполнить конфигурирование ядра, поскольку установленные обновления скорее всего окажутся выключенными. В принципе, можно подождать с компиляцией ядра до тех пор пока вы не закончите установку **iptables**.

Продолжая сборку **iptables**, запустите команду:

```
make KERNEL_DIR=/usr/src/linux/
```

Если в процессе сборки возникли какие либо проблемы, то можете попытаться разрешить их самостоятельно, либо обратиться на Netfilter mailing list, где вам смогут помочь. Там вы найдете пояснения, что могло быть сделано вами неправильно при установке, так что сразу не паникуйте. Если это не помогло - постараитесь поразмыслить логически, возможно это поможет. Или обратитесь к знакомому "гуру".

Если все прошло гладко, то следовательно вы готовы к установке исполняемых модулей (binaries), для чего запустите следующую команду:

```
make install KERNEL_DIR=/usr/src/linux/
```

Надеюсь, что здесь-то проблем не возникло! Теперь для использования пакета **iptables** вам определенно потребуется пересобрать и переустановить ядро, если вы до сих пор этого не сделали. Дополнительную информацию по установке пакета вы найдете в файле **INSTALL**.

2.3.2. Установка в Red Hat 7.1

RedHAt 7.1, с установленным ядром 2.4.x уже включает предустановленные **netfilter** и **iptables**. Однако, для сохранения обратной совместимости с предыдущими дистрибутивами, по умолчанию работает пакет ipchains. Сейчас мы коротко разберем - как удалить ipchains и запустить вместо него iptables.

Версия iptables в Red Hat 7.1 сильно устарела и, наверное неплохим решением будет установить более новую версию.

Для начала нужно отключить **ipchains**, чтобы предотвратить загрузку соответствующих модулей в будущем. Чтобы добиться этого, нам потребуется изменить имена некоторых файлов в дереве каталогов **/etc/rc.d/**. Следующая команда, выполнит требуемые действия:

```
chkconfig --level 0123456 ipchains off
```

В результате выполнения этой команды, в некоторых именах ссылок, указывающих на файлы в каталоге **/etc/rc.d/init.d/ipchains**, символ S (который сообщает, что данный сценарий отрабатывает на запуске системы) будет заменен символом K (от слова Kill, который указывает на то, что сценарий отрабатывает, при завершении работы системы. Таким образом мы предотвратим запуск ненужного сервиса в будущем.

Однако **ipchains** по-прежнему остаются в работе. Теперь надо выполнить команду, которая остановит этот сервис:

```
service ipchains stop
```

И в заключение необходимо запустить сервис **iptables**. Для этого, во-первых, надо определиться с уровнями запуска операционной системы, на которых нужно стартовать этот сервис. Обычно это уровни 2, 3 и 5. Об этих уровнях мы знаем:

- 2. Многопользовательский режим без поддержки NFS или то же самое, что и 3, но без сетевой поддержки.
- 3. Полнофункциональный многопользовательский режим.
- 5. X11. Данный уровень используется для автоматической загрузки Xwindows.

Чтобы запустить iptables на этих уровнях нужно выполнить команду:

```
chkconfig --level 235 iptables on
```

Хочется упомянуть об уровнях, на которых не требуется запуска iptables: Уровень 1 - однопользовательский режим работы, как правило используется в экстренных случаях, когда мы "поднимаем" "упавшую" систему. Уровень 4 - вообще не должен использоваться. Уровень выполнения 6 - это уровень остановки системы при выключении или перезагрузке компьютера.

Для активации сервиса iptables подадим команду:

```
service iptables start
```

Итак, мы запустили iptables, но у нас пока еще нет ни одного правила. Чтобы добавить новые правила в Red Hat 7.1 можно пойти двумя путями, во-первых: подправить файл `/etc/rc.d/init.d/iptables`, но этот способ имеет одно негативное свойство - при обновлении iptables из RPM-пакетов все ваши правила будут утеряны, а во-вторых: занести правила и сохранить их командой `iptables-save`, сохраненные таким образом правила будут автоматически восстанавливаться при загрузке системы.

В случае, если вы избрали первый вариант установки правил в iptables, то вам необходимо занести их в секцию `start` сценария `/etc/rc.d/init.d/iptables` (для установки правил при загрузке системы) или в функцию `start()`. Для выполнения действий при остановке системы - внесите соответствующие изменения в секцию `stop()` или в функцию `stop()`. Так же не забудьте про секции `restart` и `condrestart`. Хочется еще раз напомнить, что в случае обновления iptables из RPM-пакетов или через автоматическое обновление по сети, вы можете утерять все изменения, внесенные в файл `/etc/rc.d/init.d/iptables`.

Второй способ загрузки правил предпочтительнее. Он предполагает следующие шаги. Для начала - запишите правила в файл или непосредственно, через команду `iptables`, смотря что для вас предпочтительнее. Затем исполните команду `iptables-save`. Эта команда эквивалентна команде `iptables-save > /etc/sysconfig/iptables`. В результате, весь набор правил будет сохранен в файле `/etc/sysconfig/iptables`, который автоматически подгружается при запуске сервиса `iptables`. Другим способом сохранить набор правил будет подача команды `service iptables save`, которая полностью идентична вышеприведенной команде. Впоследствии, при перезагрузке компьютера, сценарий `iptables` из `rc.d` будет выполнять команду `iptables-restore` для загрузки набора правил из файла `/etc/sysconfig/iptables`.

И наконец, в завершение установки, неплохо было бы удалить старые версии `ipchains` и `iptables`. Это необходимо сделать для того, чтобы система не "перепутала" старый пакет `iptables` с новым установленным. Удаление старого пакета `iptables` необходимо произвести только в том случае, если вы производили установку из исходных текстов. Дело в том, что RPM пакеты устанавливаются в несколько иное место нежели пакеты, собранные из исходных текстов, а поэтому новый пакет не "затирает" старый. Чтобы выполнить deinсталляцию предыдущей версии `iptables` выполните следующую команду:

```
rpm -e iptables
```

Аналогичным образом удалим и `ipchains`, поскольку оставлять этот пакет в системе более нет никакого смысла.

```
rpm -e ipchains
```

Глава 3. Порядок прохождения таблиц и цепочек.

В этой главе мы рассмотрим порядок прохождения таблиц и цепочек в каждой таблице. Эта информация будет очень важна для вас позднее, когда вы начнете строить свои наборы правил, особенно когда в наборы правил будут включаться такие действия как DNAT, SNAT и конечно же TOS.

3.1. Общие положения

Когда пакет приходит на наш брандмауэр, то он сперва попадает на сетевое устройство, перехватывается соответствующим драйвером и далее передается в ядро. Далее пакет проходит ряд таблиц и затем передается либо локальному приложению, либо переправляется на другую машину. Порядок следования пакета приводится ниже: **Таблица 3-1. Порядок движения транзитных пакетов** Шаг Таблица Цепочка Примечание

1 Кабель (т.е. Интернет)

2 Сетевой интерфейс (например, `eth0`)

3 mangle PREROUTING Обычно эта цепочка используется для внесения изменений в заголовок пакета, например для изменения битов TOS и пр..

4 nat PREROUTING Эта цепочка используется для трансляции сетевых адресов (*Destination Network Address Translation*). Source Network Address Translation выполняется позднее, в другой цепочке. Любой рода фильтрация в этой цепочке может производиться только в исключительных случаях

5 Принятие решения о дальнейшей маршрутизации, т.е. в этой точке решается куда пойдет пакет - локальному приложению или на другой узел сети.

6 mangle FORWARD Далее пакет попадает в цепочку FORWARD таблицы mangle, которая должна использоваться только в исключительных случаях, когда необходимо внести некоторые изменения в заголовок пакета между двумя точками принятия решения о маршрутизации.

7 Filter FORWARD В цепочку FORWARD попадают только те пакеты, которые идут на другой хост Вся фильтрация транзитного трафика должна выполняться здесь. Не забывайте, что через эту цепочку проходит трафик в обоих направлениях, обязательно учитывайте это обстоятельство при написании правил фильтрации.

8 mangle POSTROUTING Эта цепочка предназначена для внесения изменений в заголовок пакета уже после того как принято последнее решение о маршрутизации.

9 nat POSTROUTING Эта цепочка предназначена в первую очередь для Source Network Address Translation. Не используйте ее для фильтрации без особой необходимости. Здесь же выполняется и маскарадинг (Masquerading).

10 Выходной сетевой интерфейс (например, eth1).

11 Кабель (пусть будет LAN).

Как вы можете видеть, пакет проходит несколько этапов, прежде чем он будет передан далее. На каждом из них пакет может быть остановлен, будь то цепочка iptables или что либо еще, но нас главным образом интересует iptables. Заметьте, что нет каких либо цепочек, специфичных для отдельных интерфейсов или чего либо подобного. Цепочку FORWARD проходят ВСЕ пакеты, которые движутся через наш брандмауэр/роутер. Не используйте цепочку INPUT для фильтрации транзитных пакетов, они туда просто не попадают! Через эту цепочку движутся только те пакеты, которые предназначены данному хосту!

А теперь рассмотрим порядок движения пакета, предназначенного локальному процессу/приложению:

Таблица 3-2. Для локального приложения Шаг Таблица Цепочка Примечание

1 Кабель (т.е. Интернет)

2 Входной сетевой интерфейс (например, eth0)

3 mangle PREROUTING Обычно используется для внесения изменений в заголовок пакета, например для установки битов TOS и пр.

4 nat PREROUTING Преобразование адресов (*Destination Network Address Translation*). Фильтрация пакетов здесь допускается только в исключительных случаях.

5 Принятие решения о маршрутизации.

6 mangle INPUT Пакет попадает в цепочку INPUT таблицы mangle. Здесь внесются изменения в заголовок пакета перед тем как он будет передан локальному приложению.

7 filter INPUT Здесь производится фильтрация входящего трафика. Помните, что все входящие пакеты, адресованные нам, проходят через эту цепочку, независимо от того с какого интерфейса они поступили.

8 Локальный процесс/приложение (т.е., программа-сервер или программа-клиент)

Важно помнить, что на этот раз пакеты идут через цепочку INPUT, а не через FORWARD.

И в заключение мы рассмотрим порядок движения пакетов, созданных локальными процессами.

Таблица 3-3. От локальных процессов Шаг Таблица Цепочка Примечание

1 Локальный процесс (т.е., программа-сервер или программа-клиент).

2 Принятие решения о маршрутизации. Здесь решается куда пойдет пакет дальше -- на какой адрес, через какой сетевой интерфейс и пр.

3 mangle OUTPUT Здесь производится внесение изменений в заголовок пакета. Выполнение фильтрации в этой цепочке может иметь негативные последствия.

4 nat OUTPUT Эта цепочка используется для трансляции сетевых адресов (NAT) в пакетах, исходящих от локальных процессов брандмауэра.

5 Filter OUTPUT Здесь фильтруется исходящий трафик.

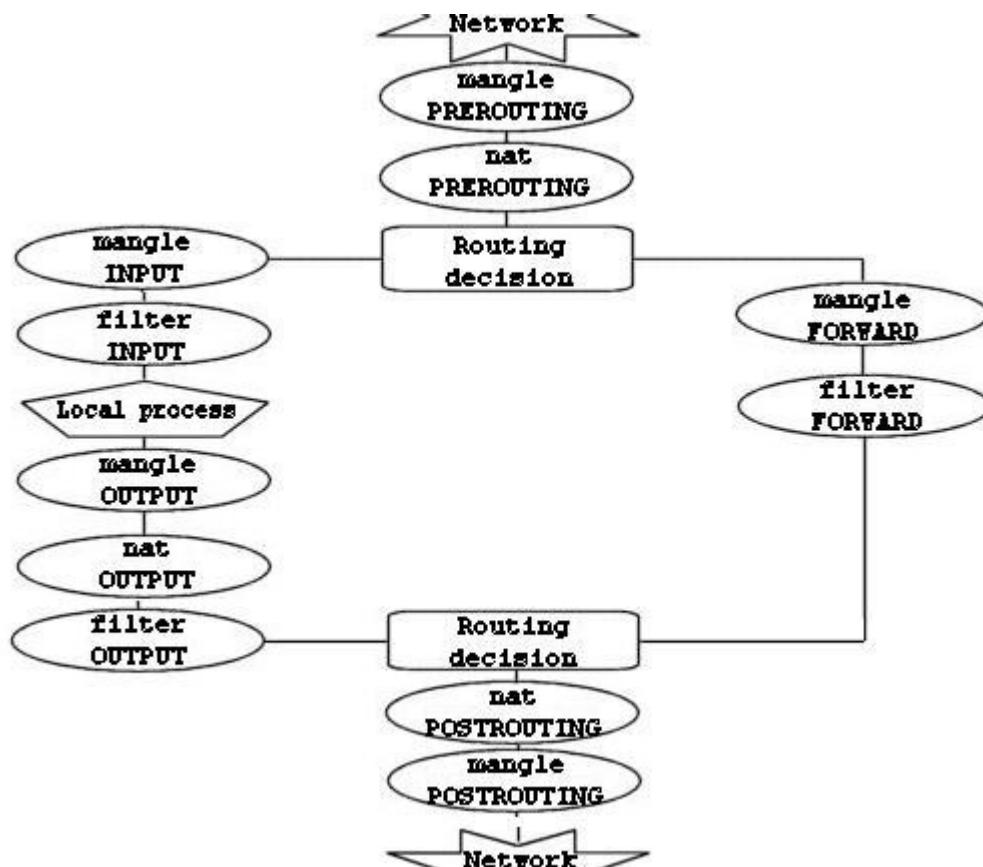
6 mangle POSTROUTING Цепочка POSTROUTING таблицы mangle в основном используется для правил, которые должны вносить изменения в заголовок пакета перед тем, как он покинет брандмауэр, но уже после принятия решения о маршрутизации. В эту цепочку попадают все пакеты, как транзитные, так и созданные локальными процессами брандмауэра.

7 nat POSTROUTING Здесь выполняется Source Network Address Translation. Не следует в этой цепочке производить фильтрацию пакетов во избежание нежелательных побочных эффектов. Однако и здесь можно останавливать пакеты, применяя политику по-умолчанию DROP.

8 Сетевой интерфейс (например, eth0)

9 Кабель (т.е., Internet)

Теперь мы знаем, что есть три различных варианта прохождения пакетов. Рисунок ниже более наглядно демонстрирует это:



Этот рисунок дает довольно ясное представление о порядке прохождения пакетов через различные цепочки. В первой точке принятия решения о маршрутизации (routing decision) все пакеты, предназначенные данному хосту направляются в цепочку INPUT, остальные - в цепочку FORWARD.

Обратите внимание также на тот факт, что пакеты, с адресом назначения на брандмауэр, могут претерпеть изменение сетевого адреса назначения (DNAT) в цепочке PREROUTING таблицы nat и соответственно дальнейшая маршрутизация в первой точке будет выполняться в зависимости от произведенных изменений. Запомните - все пакеты проходят через таблицы и цепочки по тому или иному маршруту. Даже если выполняется DNAT в ту же сеть, откуда пакет пришел, то он все равно продолжит движение по цепочкам.

В сценарии `rc.test-iptables.txt` вы сможете найти дополнительную информацию о порядке прохождения пакетов.

3.2. Таблица Mangle

Как уже упоминалось выше, эта таблица предназначена, главным образом для внесения изменений в заголовки пакетов (mangle - искажать, изменять. прим. перев.). Т.е. в этой таблице вы можете устанавливать биты TOS (Type Of Service) и т.д.

Еще раз напоминаю вам, что в этой таблице не следует производить любого рода фильтрацию, маскировку или преобразование адресов (DNAT, SNAT, MASQUERADE).

В этой таблице допускается выполнять только нижеперечисленные действия:

- TOS
- TTL
- MARK

Действие **TOS** выполняет установку битов поля Type of Service в пакете. Это поле используется для назначения сетевой политики обслуживания пакета, т.е. задает желаемый вариант маршрутизации. Однако, следует заметить, что данное свойство в действительности используется на незначительном количестве маршрутизаторов в Интернете. Другими словами, не следует изменять состояние этого поля для пакетов, уходящих в Интернет, потому что на роутерах, которые таки обслуживают это поле, может быть принято неправильное решение при выборе маршрута.

Действие **TTL** используется для установки значения поля TTL (Time To Live) пакета. Есть одно неплохое применение этому действию. Мы можем присваивать определенное значение этому полю, чтобы скрыть наш брандмауэр от чересчур любопытных провайдеров (Internet Service Providers). Дело в том, что отдельные провайдеры очень не любят когда одно подключение разделяется несколькими компьютерами. и тогда они начинают проверять значение TTL приходящих пакетов и используют его как один из критериев определения того, один компьютер "сидит" на подключении или несколько.

Действие **MARK** устанавливает специальную метку на пакет, которая затем может быть проверена другими правилами в iptables или другими программами, например iproute2. С помощью "меток" можно управлять маршрутизацией пакетов, ограничивать трафик и т.п.

3.3. Таблица Nat

Эта таблица используется для выполнения преобразований сетевых адресов NAT (Network Address Translation). Как уже упоминалось ранее, только первый пакет из потока проходит через цепочки этой таблицы, трансляция адресов или маскировка применяются ко всем последующим пакетам в потоке автоматически. Для этой таблицы характерны действия:

- DNAT
- SNAT
- MASQUERADE

Действие **DNAT** (Destination Network Address Translation) производит преобразование адресов назначения в заголовках пакетов. Другими словами, этим действием производится перенаправление пакетов на другие адреса, отличные от указанных в заголовках пакетов.

SNAT (Source Network Address Translation) используется для изменения исходных адресов пакетов. С помощью этого действия можно скрыть структуру локальной сети, а заодно и разделить единственный внешний IP адрес между компьютерами локальной сети для выхода в Интернет. В этом случае брандмауэр, с помощью SNAT, автоматически производит прямое и обратное преобразование адресов, тем самым давая возможность выполнять подключение к серверам в Интернете с компьютеров в локальной сети.

Маскировка (MASQUERADE) применяется в тех же целях, что и SNAT, но в отличие от последней, MASQUERADE дает более сильную нагрузку на систему. Происходит это потому, что каждый раз, когда требуется выполнение этого действия - производится запрос IP адреса для указанного в действии сетевого интерфейса, в то время как для SNAT IP адрес указывается непосредственно. Однако, благодаря такому отличию, MASQUERADE может работать в случаях с динамическим IP адресом, т.е. когда вы подключаетесь к Интернет, скажем через PPP, SLIP или DHCP.

3.4. Таблица Filter

Как следует из названия, в этой таблице должны содержаться наборы правил для выполнения фильтрации пакетов. Пакеты могут пропускаться далее, либо отвергаться (действия ACCEPT и DROP соответственно), в зависимости от их содержимого. Конечно же, мы можем отфильтровывать пакеты и в других таблицах, но эта таблица существует именно для нужд фильтрации. В этой таблице допускается использование большинства из существующих действий, однако ряд действий, которые мы рассмотрели выше в этой главе, должны выполняться только в присущих им таблицах.

Глава 4. Механизм определения состояний

В данной главе все внимание будет уделено механизму определения состояний пакетов (state machine). По прочтении ее у вас должно сложиться достаточно четкое представление о работе механизма, а способствовать этому должен значительный объем поясняющих примеров.

4.1. Введение

Механизм определения состояния (state machine) является отдельной частью iptables и в действительности не должен бы так называться, поскольку фактически является механизмом трассировки соединений. Однако значительному количеству людей он известен именно как "механизм определения состояния" (state machine). В данной главе эти названия будут использоваться как синонимы. Трассировщик соединений создан для того, чтобы netfilter мог постоянно иметь информацию о состоянии каждого конкретного соединения. Наличие трассировщика позволяет создавать более надежные наборы правил по сравнению с брандмауэрами, которые не имеют поддержки такого механизма.

В пределах iptables, соединение может иметь одно из 4-х базовых состояний: NEW, ESTABLISHED, RELATED и INVALID. Позднее мы остановимся на каждом из них более подробно. Для управления прохождением пакетов, основываясь на их состоянии, используется критерий --state.

Трассировка соединений производится специальным кодом в пространстве ядра -- трассировщиком (conntrack). Код трассировщика может быть скомпилирован как подгружаемый модуль ядра, так и статически связан с ядром. В большинстве случаев нам потребна более специфичная информация о соединении, чем та, которую поставляет трассировщик по-умолчанию. Поэтому трассировщик включает в себя обработчики различных протоколов, например TCP, UDP или ICMP. Собранная ими информация затем используется для идентификации и определения текущего состояния соединения. Например - соединение по протоколу UDP однозначно идентифицируется по IP-адресам и портам источника и приемника.

В предыдущих версиях ядра имелась возможность включения/выключения поддержки дефрагментации пакетов. Однако, после того как трассировка соединений была включена в состав iptables/netfilter, надобность в этом отпала. Причина в том, что трассировщик не в состоянии выполнять возложенные на него функции без поддержки дефрагментации и поэтому она включена постоянно. Ее нельзя отключить иначе как отключив трассировку соединений. Дефрагментация выполняется всегда, если трассировщик включен.

Трассировка соединений производится в цепочке **PREROUTING**, исключая случаи, когда пакеты создаются локальными процессами на брандмауэре, в этом случае трассировка производится в цепочке **OUTPUT**. Это означает, что iptables производит все вычисления, связанные с определением состояния, в пределах этих цепочек. Когда локальный процесс на брандмауэре отправляет первый пакет из потока, то в цепочке **OUTPUT** ему присваивается состояние **NEW**, а когда возвращается пакет ответа, то состояние соединения в цепочке **PREROUTING** изменяется на **ESTABLISHED**, и так далее. Если же соединение устанавливается извне, то состояние **NEW** присваивается первому пакету из потока в цепочке **PREROUTING**. Таким образом, определение состояния пакетов производится в пределах цепочек **PREROUTING** и **OUTPUT** таблицы **nat**.

4.2. Таблица трассировщика

Кратко рассмотрим таблицу трассировщика, которую можно найти в файле `/proc/net/ip_conntrack`. Здесь содержится список всех активных соединений. Если модуль `ip_conntrack` загружен, то команда `cat /proc/net/ip_conntrack` должна вывести нечто, подобное:

```
tcp 6 117 SYN_SENT src=192.168.1.6 dst=192.168.1.9 sport=32775  
dport=22 [UNREPLIED] src=192.168.1.9 dst=192.168.1.6 sport=22  
dport=32775 use=2
```

В этом примере содержится вся информация, которая известна трассировщику, по конкретному соединению. Первое, что можно увидеть - это название протокола, в данном случае - `tcp`. Далее следует некоторое число в обычном десятичном представлении. После него следует число, определяющее "время жизни" записи в таблице (т.е. количество секунд, через которое информация о соединении будет удалена из таблицы). Для нашего случая, запись в таблице будет храниться еще 117 секунд, если конечно через это соединение более не проследует ни одного пакета. При прохождении каждого последующего пакета через данное соединение, это значение будет устанавливаться в значение по-умолчанию для заданного состояния. Это число уменьшается на 1 каждую секунду. Далее следует фактическое состояние соединения. Для нашего примера состояние имеет значение `SYN_SENT`. Внутреннее представление состояния несколько отличается от внешнего. Значение `SYN_SENT` говорит о том, что через данное соединение проследовал единственный пакет `TCP SYN`. Далее расположены адреса отправителя и получателя, порт отправителя и получателя. Здесь же видно ключевое слово `[UNREPLIED]`, которое сообщает о том, что ответного трафика через это соединение еще не было. И наконец приводится дополнительная информация по ожидаемому пакету, это IP адреса отправителя/получателя (те же самые, только поменявшиеся местами, поскольку ожидается ответный пакет), то же касается и портов.

Записи в таблице могут принимать ряд значений, все они определены в заголовочных файлах `linux/include/netfilter-ipv4/ip_conntrack*.h`. Значения по-умолчанию зависят от типа протокола. Каждый из IP-протоколов - `TCP`, `UDP` или `ICMP` имеют собственные значения по-умолчанию, которые определены в заголовочном файле `linux/include/netfilter-ipv4/ip_conntrack.h`. Более подробно мы остановимся на этих значениях, когда будем рассматривать каждый из протоколов в отдельности.

Совсем недавно, в `patch-o-matic`, появилась заплата `tcp-window-tracking`, которая предоставляет возможность передачи значений всех таймаутов через специальные переменные, т.е. позволяет изменять их "на лету". Таким образом появляется возможность изменения таймаутов без необходимости пересборки ядра. Изменения вносятся с помощью определенных системных вызовов, через каталог `/proc/sys/net/ipv4/netfilter`. Особое внимание обратите на ряд переменных `/proc/sys/net/ipv4/netfilter/ip_ct_*`.

После получения пакета ответа трассировщик снимет флаг [**UNREPLIED**] и заменит его флагом [**ASSURED**]. Этот флаг сообщает о том, что соединение установлено уверенно и эта запись не будет стерта по достижении максимально возможного количества трассируемых соединений. Максимальное количество записей, которое может содержаться в таблице зависит от значения по-умолчанию, которое может быть установлено вызовом функции `ipsysctl` в последних версиях ядра. Для объема ОЗУ 128 Мб это значение соответствует 8192 записям, для 256 Мб - 16376. Вы можете посмотреть и изменить это значение установкой переменной `/proc/sys/net/ipv4/ip_conntrack_max`.

4.3. Состояния в пространстве пользователя

Как вы уже наверняка заметили, в пространстве ядра, в зависимости от типа протокола, пакеты могут иметь несколько различных состояний. Однако, вне ядра пакеты могут иметь только 4 состояния. В основном состояние пакета используется критерием `--state`. Допустимыми являются состояния **NEW**, **ESTABLISHED**, **RELATED** и **INVALID**. В таблице, приводимой ниже, рассматриваются каждое из возможных состояний.

Таблица 4-1. Перечень состояний в пространстве пользователя Состояние Описание

NEW Признак **NEW** сообщает о том, что пакет является первым для данного соединения. Это означает, что это первый пакет в данном соединении, который увидел модуль трассировщика. Например если получен SYN пакет являющийся первым пакетом для данного соединения, то он получит статус **NEW**. Однако, пакет может и не быть SYN пакетом и тем не менее получить статус **NEW**. Это может породить определенные проблемы в отдельных случаях, но может оказаться и весьма полезным, например когда желательно "подхватить" соединения, "потерянные" другими брандмауэрами или в случаях, когда таймаут соединения уже истек, но само соединение не было закрыто.

RELATED Состояние **RELATED** одно из самых "хитрых". Соединение получает статус **RELATED** если оно связано с другим соединением, имеющим признак **ESTABLISHED**. Это означает, что соединение получает признак **RELATED** тогда, когда оно инициировано из уже установленного соединения, имеющего признак **ESTABLISHED**. Хорошим примером соединения, которое может рассматриваться как **RELATED**, является соединение FTP-data, которое является связанным с портом FTP control, а так же DCC соединение, запущенное из IRC. Обратите внимание на то, что большинство протоколов TCP и некоторые из протоколов UDP весьма сложны и передают информацию о соединении через область данных TCP или UDP пакетов и поэтому требуют наличия специальных вспомогательных модулей для корректной работы.

ESTABLISHED Состояние **ESTABLISHED** говорит о том, что это не первый пакет в соединении. Схема установки состояния **ESTABLISHED** достаточна проста для понимания. Единственное требование, предъявляемое к соединению, заключается в том, что для перехода в состояние **ESTABLISHED** необходимо чтобы узел сети передал пакет и получил на него ответ от другого узла (хоста). После получения ответа состояние соединения **NEW** или **RELATED** будет изменено на **ESTABLISHED**.

INVALID Признак **INVALID** говорит о том, что пакет не может быть идентифицирован и поэтому не может иметь определенного статуса. Это может происходить по разным причинам, например при нехватке памяти или при получении ICMP-сообщения об ошибке, которое не соответствует какому либо известному соединению. Наверное наилучшим вариантом было бы применение действия **DROP** к таким пакетам.

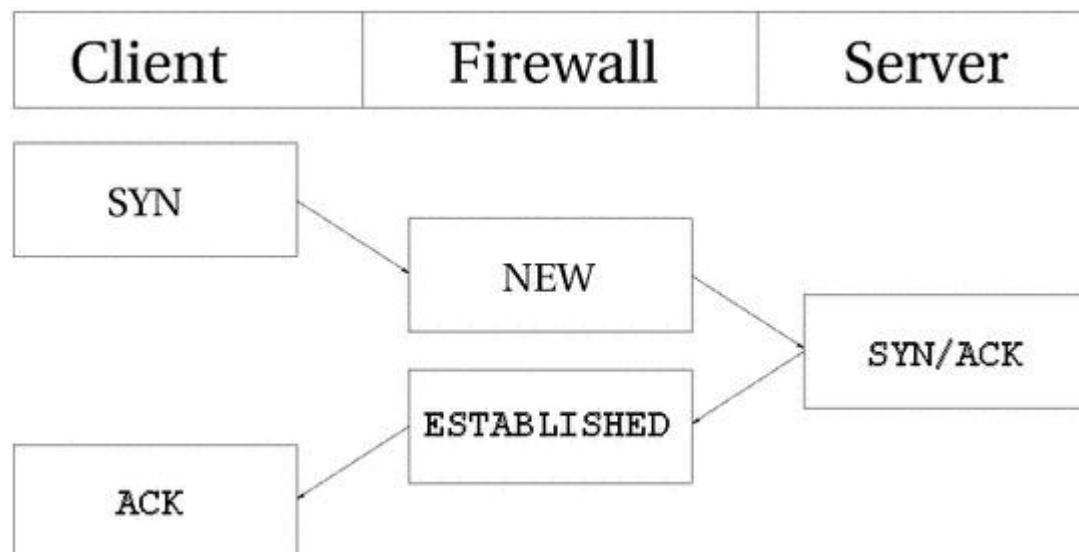
Эти четыре состояния могут использоваться в критерии `--state`. Механизм определения состояния позволяет строить чрезвычайно мощную и эффективную защиту. Раньше приходилось открывать все порты выше 1024, чтобы пропустить обратный трафик в локальную сеть, теперь же, при наличии механизма определения состояния, необходимость в этом отпала, поскольку появилась возможность "открывать" доступ только для обратного (ответного) трафика, пресекая попытки установления соединений извне.

4.4. TCP соединения

В этом и в последующих разделах мы поближе рассмотрим признаки состояний и порядок их обработки каждым из трех базовых протоколов TCP, UDP и ICMP, а так же коснемся случая, когда протокол соединения не может быть классифицирован на принадлежность к трем, вышеуказанным, протоколам. Начнем рассмотрение с протокола TCP, поскольку он имеет множество интереснейших особенностей в отношении механизма определения состояния в iptables.

TCP соединение всегда устанавливается передачей трех пакетов, которые инициализируют и устанавливают соединение, через которое в дальнейшем будут передаваться данные. Сессия начинается с передачи SYN пакета, в ответ на который передается SYN/ACK пакет и подтверждает установление соединения пакет ACK. После этого соединение считается установленным и готовым к передаче данных. Может возникнуть вопрос: "А как же трассируется соединение?". В действительности все довольно просто.

Для всех типов соединений, трассировка проходит практически одинаково. Взгляните на рисунок ниже, где показаны все стадии установления соединения. Как видите, трассировщик, с точки зрения пользователя, фактически не следит за ходом установления соединения. Просто, как только трассировщик "увидел" первый (SYN) пакет, то присваивает ему статус NEW. Как только через трассировщика проходит второй пакет (SYN/ACK), то соединению присваивается статус ESTABLISHED. Почему именно второй пакет? Сейчас разберемся. Столя свой набор правил, вы можете позволить покидать локальную сеть пакетам со статусом NEW и ESTABLISHED, а во входящем трафике пропускать пакеты только со статусом ESTABLISHED и все будет работать прекрасно. И наоборот, если бы трассировщик продолжал считать соединение как NEW, то фактически вам никогда не удалось бы установить соединение с "внешним миром", либо пришлось бы позволить прохождение NEW пакетов в локальную сеть. С точки зрения ядра все выглядит более сложным, поскольку в пространстве ядра TCP соединения имеют ряд промежуточных состояний, недоступных в пространстве пользователя. В общих чертах они соответствуют спецификации RFC 793 - Transmission Control Protocol на странице 21-23. Более подробно эта тема будет рассматриваться чуть ниже.



С точки зрения пользователя все выглядит достаточно просто, однако если посмотреть с точки зрения ядра, то все выглядит несколько сложнее. Рассмотрим порядок изменения состояния соединения в таблице /proc/net/ip_conntrack. После передачи первого пакета SYN.

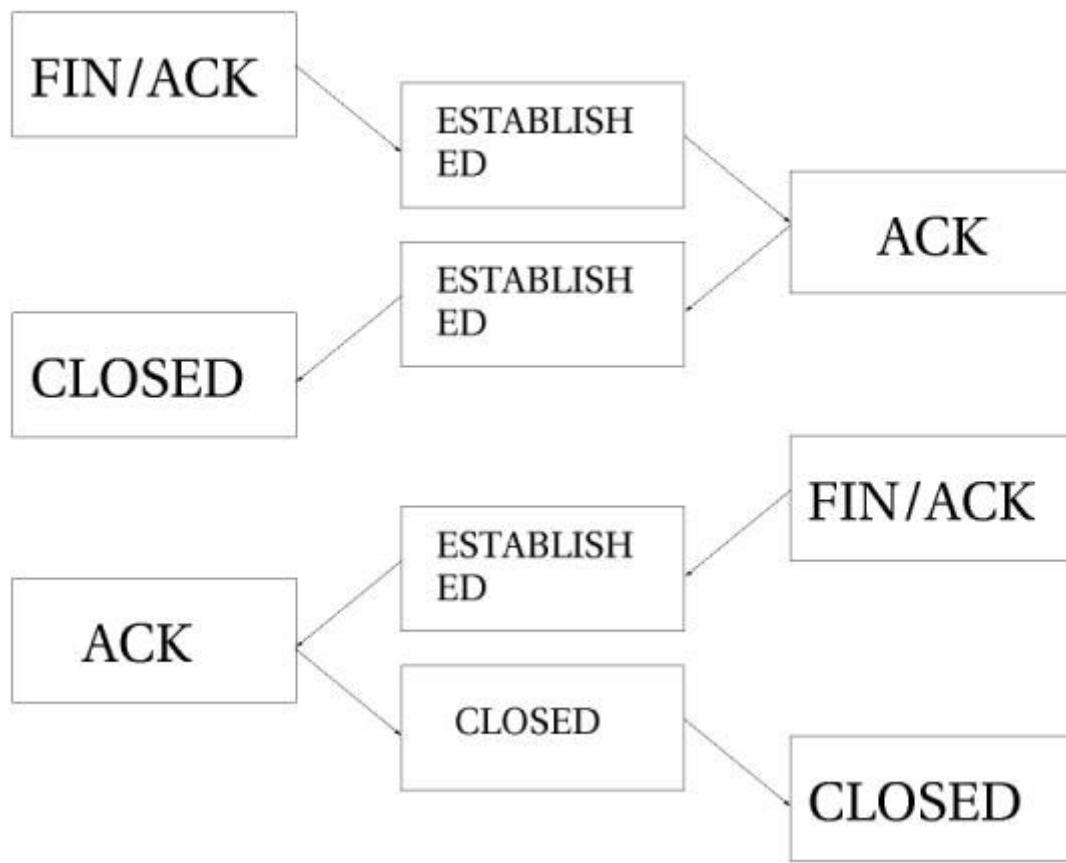
```
tcp      6 117 SYN_SENT src=192.168.1.5 dst=192.168.1.35 sport=1031 \
          dport=23 [UNREPLIED] src=192.168.1.35 dst=192.168.1.5 sport=23 \
          dport=1031 use=1
```

Теперь запись сообщает о том, что обратно прошел пакет SYN/ACK. На этот раз соединение переводится в состояние SYN_RECV. Это состояние говорит о том, что пакет SYN был благополучно доставлен получателю и в ответ на него пришел пакет-подтверждение (SYN/ACK). Кроме того, механизм определения состояния "увидев" пакеты, следующие в обеих направлениях, снимает флаг [UNREPLIED]. И наконец после передачи заключительного ACK-пакета, в процедуре установления соединения

```
tcp      6 431999 ESTABLISHED src=192.168.1.5 dst=192.168.1.35 \
        sport=1031 dport=23 src=192.168.1.35 dst=192.168.1.5 \
        sport=23 dport=1031 use=1
```

соединение переходит в состояние ESTABLISHED (установленное). После приема нескольких пакетов через это соединение, к нему добавится флаг [ASSURED] (уверенное).

При закрытии, TCP соединение проходит через следующие состояния:



Как видно из рисунка, соединение не закрывается до тех пор пока не будет передан последний пакет ACK. Обратите внимание - эта картинка описывает нормальный процесс закрытия соединения. Кроме того, если соединение отвергается, то оно может быть закрыто передачей пакета RST (сброс). В этом случае соединение будет закрыто по истечению предопределенного времени.

При закрытии, соединение переводится в состояние TIME_WAIT, продолжительность которого по-умолчанию соответствует 2 минутам, в течение которого еще возможно прохождение пакетов через брандмауэр. Это является своего рода "буферным временем", которое дает возможность пройти пакетам, "увязшим" на том или ином маршрутизаторе (роутере).

Если соединение закрывается по получении пакета RST, то оно переводится в состояние CLOSE. Время ожидания до фактического закрытия соединения по-умолчанию устанавливается равным 10 секунд. Подтверждение на пакеты RST не передается и соединение закрывается сразу же. Кроме того имеется ряд других внутренних состояний. В таблице ниже приводится список возможных внутренних состояний соединения и соответствующие им размеры таймаутов.

Таблица 4-2. Internal states Состояние Время ожидания

NONE

ESTABLISHED 5 дней

SYN_SENT 2 минуты

SYN_RECV 60 секунд

FIN_WAIT 2 минуты

TIME_WAIT 2 минуты

CLOSE 10 секунд

CLOSE_WAIT 12 часов

LAST_ACK 30 секунд

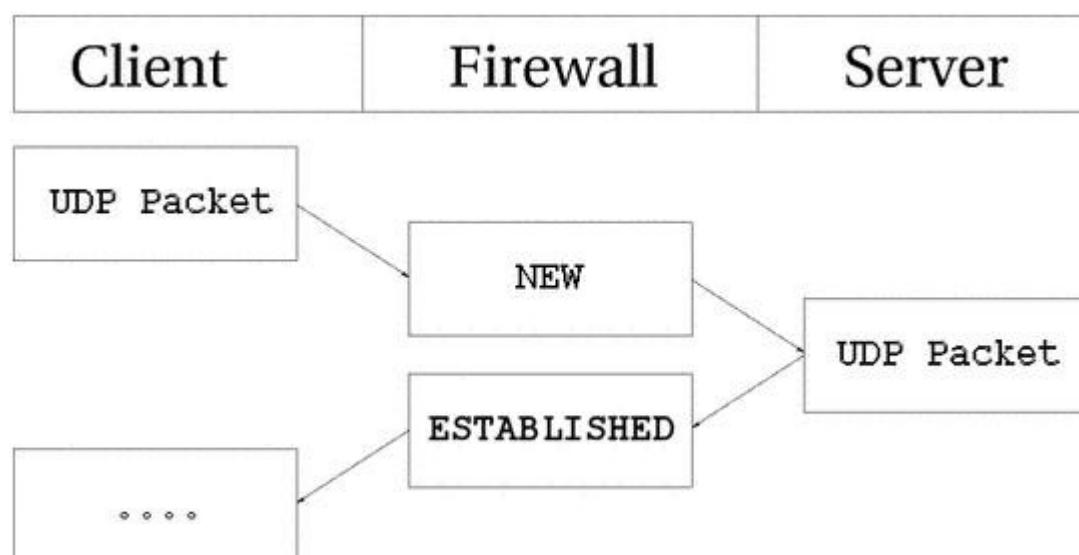
LISTEN> 2 минуты

Эти значения могут несколько изменяться от версии к версии ядра, кроме того, они могут быть изменены через интерфейс файловой системы /proc (переменные proc/sys/net/ipv4/netfilter/ip_ct_tcp_*). Значения устанавливаются в сотых долях секунды, так что число 3000 означает 30 секунд.

Обратите внимание на то, что со стороны пользователя, механизм определения состояния никак не отображает состояние флагов TCP пакетов. Как правило - это не всегда хорошо, поскольку состояние NEW присваивается, не только пакетам SYN. Это качество трассировщика может быть использовано для избыточного файерволлинга (firewalling), но для случая домашней локальной сети, в которой используется только один брандмауэр это очень плохо. Эта проблема более подробно обсуждается в разделе *Пакеты со статусом NEW и со сброшенным битом SYN* приложения *Общие проблемы и вопросы*. Альтернативным вариантом решения этой проблемы может служить установка заплаты tcp-window-tracking из patch-o-matic, которая сделает возможным принятие решений в зависимости от значения TCP window.

4.5. UDP соединения

По сути своей, UDP соединения не имеют признака состояния. Этому имеется несколько причин, основная из них состоит в том, что этот протокол не предусматривает установления и закрытия соединения, но самый большой недостаток - отсутствие информации об очередности поступления пакетов. Приняв две датаграммы UDP, невозможно сказать точно в каком порядке они были отправлены. Однако, даже в этой ситуации все еще возможно определить состояние соединения. Ниже приводится рисунок того, как выглядит установление соединения с точки зрения трассировщика.



Из рисунка видно, что состояние UDP соединения определяется почти так же как и состояние TCP соединения, с точки зрения из пользовательского пространства. Изнутри же это выглядит несколько иначе, хотя во многом похоже. Для начала посмотрим на запись, появившуюся после передачи первого пакета UDP.

```
udp      17 20 src=192.168.1.2 dst=192.168.1.5 sport=137 dport=1025 \
[UNREPLIED] src=192.168.1.5 dst=192.168.1.2 sport=1025 \
dport=137 use=1
```

Первое, что мы видим -- это название протокола (udp) и его номер (см. /etc/protocols прим. перев.). Третье значение -- оставшееся "время жизни" записи в секундах. Далее следуют характеристики пакета, прошедшего через брандмауэр -- это адреса и порты отправителя и получателя. Здесь же видно, что это первый пакет в сессии (флаг [UNREPLIED]). И завершают запись адреса и порты отправителя и получателя ожидаемого пакета. Таймаут такой записи по умолчанию составляет 30 секунд.

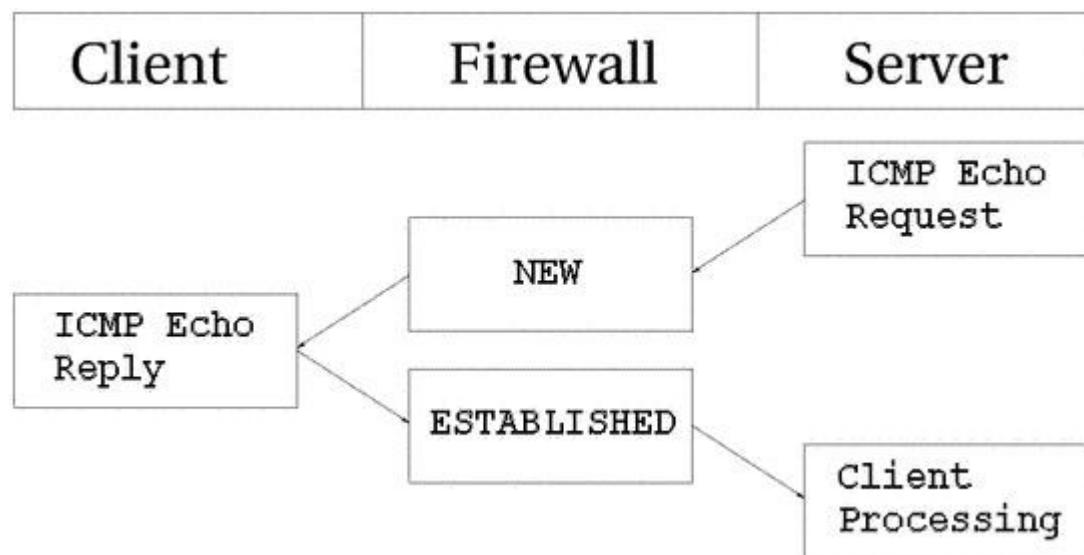
```
udp      17      170      src=192.168.1.2      dst=192.168.1.5      sport=137      \
dport=1025      src=192.168.1.5      dst=192.168.1.2      sport=1025      \
dport=137 use=1
```

После того как сервер "увидел" ответ на первый пакет, соединение считается ESTABLISHED (установленным), единственное отличие от предыдущей записи состоит в отсутствии флага [UNRREPLIED] и, кроме того, таймаут для записи стал равным 180 секундам. После этого может только добавиться флаг [ASSURED] (уверенное соединение), который был описан выше. Флаг [ASSURED] устанавливается только после прохождения некоторого количества пакетов через соединение.

```
udp      17      175      src=192.168.1.5      dst=195.22.79.2      sport=1025      \
dport=53      src=195.22.79.2      dst=192.168.1.5      sport=53      \
dport=1025 [ASSURED] use=1
```

Теперь соединение стало "уверенным". Запись в таблице выглядит практически так же как и в предыдущем примере, за исключением флага [ASSURED]. Если в течение 180 секунд через соединение не пройдет хотя бы один пакет, то запись будет удалена из таблицы. Это достаточно маленький промежуток времени, но его вполне достаточно для большинства применений. "Время жизни" отсчитывается от момента прохождения последнего пакета и при появлении нового, время переустанавливается в свое начальное значение, это справедливо и для всех остальных типов внутренних состояний.

4.6. ICMP соединения



Как видно из этого рисунка, сервер выполняет Echo Request (эхо-запрос) к клиенту, который (запрос) распознается брандмауэром как NEW. На этот запрос клиент отвечает пакетом Echo Reply, и теперь пакет распознается как имеющий состояние ESTABLISHED. После прохождения первого пакета (Echo Request) в ip_conntrack появляется запись:

```
icmp      1      25      src=192.168.1.6      dst=192.168.1.10      type=8      code=0      \
id=33029      [UNREPLIED]      src=192.168.1.10      dst=192.168.1.6      \
type=0 code=0 id=33029 use=1
```

Эта запись несколько отличается от записей, свойственных протоколам TCP и UDP, хотя точно так же присутствуют и название протокола и время таймаута и адреса передатчика и приемника, но далее появляются три новых поля - `type`, `code` и `id`. Поле `type` содержит тип ICMP, поле `code` - код ICMP. Значения типов и кодов ICMP приводятся в приложении *Типы ICMP*. И последнее поле `id` содержит идентификатор пакета. Каждый ICMP-пакет имеет свой идентификатор. Когда приемник, в ответ на ICMP-запрос посыпает ответ, он подставляет в пакет ответа этот идентификатор, благодаря чему, передатчик может корректно распознать в ответ на какой запрос пришел ответ.

Следующее поле - флаг `[UNREPLIED]`, который встречался нам ранее. Он означает, что прибыл первый пакет в соединении. Завершается запись характеристиками ожидаемого пакета ответа. Сюда включаются адреса отправителя и получателя. Что касается типа и кода ICMP пакета, то они соответствуют правильным значениям ожидаемого пакета ICMP Echo Reply. Идентификатор пакета-ответа тот же, что и в пакете запроса.

Пакет ответа распознается уже как ESTABLISHED. Однако, мы знаем, что после передачи пакета ответа, через это соединение уже ничего не ожидается, поэтому после прохождения ответа через netfilter, запись в таблице трассировщика уничтожается.

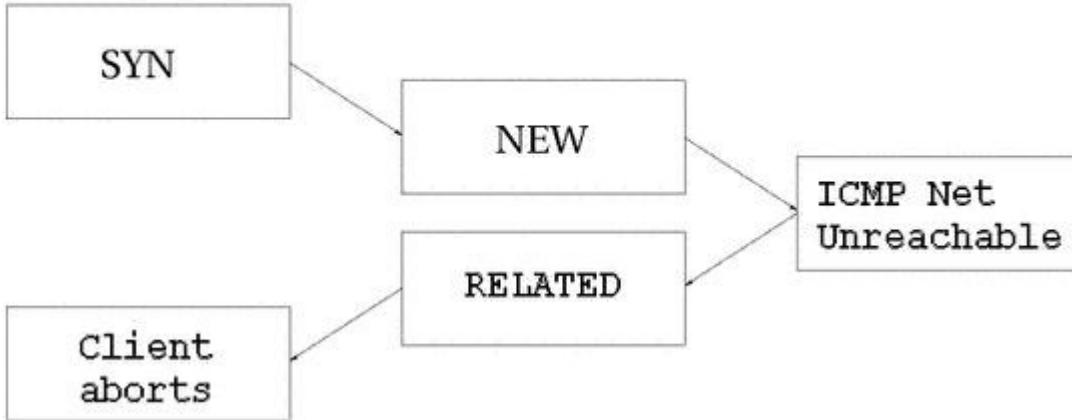
В любом случае запрос рассматривается как NEW, а ответ как ESTABLISHED.

Заметьте при этом, что пакет ответа должен совпадать по своим характеристикам (адреса отправителя и получателя, тип, код и идентификатор) с указанными в записи в таблице трассировщика, это справедливо и для всех остальных типов трафика.

ICMP запросы имеют таймаут, по-умолчанию, 30 секунд. Этого времени, в большинстве случаев, вполне достаточно. Время таймаута можно изменить в `/proc/sys/net/ipv4/netfilter/ip_ct_icmp_timeout`. (Напоминаю, что переменные типа `/proc/sys/net/ipv4/netfilter/ip_ct_*` становятся доступны только после установки "заплаты" `tcp-window-tracking` из `patch-o-matic` прим. перев.).

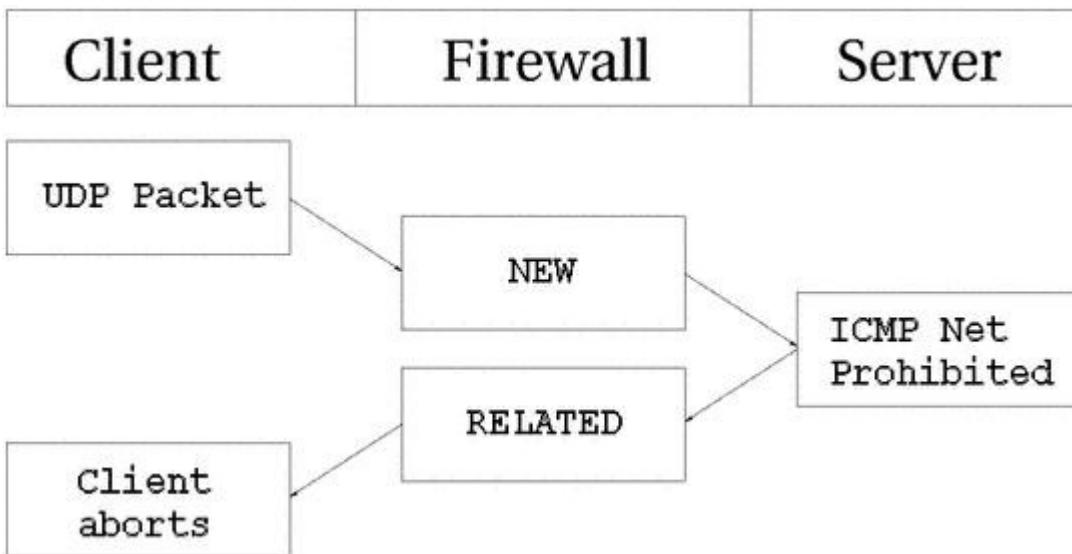
Значительная часть ICMP используется для передачи сообщений о том, что происходит с тем или иным UDP или TCP соединением. Всвязи с этим они очень часто распознаются как связанные (RELATED) с существующим соединением. Простым примером могут служить сообщения ICMP Host Unreachable или ICMP Network Unreachable. Они всегда порождаются при попытке соединиться с узлом сети когда этот узел или сеть недоступны, в этом случае последний маршрутизатор вернет соответствующий ICMP пакет, который будет распознан как RELATED. На рисунке ниже показано как это происходит.

| Client | Firewall | Server |
|--------|----------|--------|
|--------|----------|--------|



В этом примере некоторому узлу передается запрос на соединение (SYN пакет). Он приобретает статус NEW на брандмауэре. Однако, в этот момент времени, сеть оказывается недоступной, поэтому роутер возвращает пакет ICMP Network Unreachable. Трассировщик соединений распознает этот пакет как RELATED, благодаря уже имеющейся записи в таблице, так что пакет благополучно будет передан клиенту, который затем оборвёт неудачное соединение. Тем временем, брандмауэр уничтожит запись в таблице, поскольку для данного соединения было получено сообщение об ошибке.

То же самое происходит и с UDP соединениями - если обнаруживаются подобные проблемы. Все сообщения ICMP, передаваемые в ответ на UDP соединение, рассматриваются как RELATED. Взгляните на следующий рисунок.



Датаграмма UDP передается на сервер. Соединению присваивается статус NEW. Однако доступ к сети запрещен (брандмауэром или роутером), поэтому обратно возвращается сообщение ICMP Network Prohibited. Брандмауэр распознает это сообщение как связанное с открытым UDP соединением, присваивает ему статус RELATED и передает клиенту. После чего запись в таблице трассировщика уничтожается, а клиент благополучно обрывает соединение.

4.7. Поведение по-умолчанию

В некоторых случаях механизм определения состояния не может распознать протокол обмена и, соответственно, не может выбрать стратегию обработки этого соединения. В этом случае он переходит к заданному по-умолчанию поведению. Поведение по-умолчанию используется, например при обслуживании протоколов NETBLT, MUX и EGP. Поведение по-умолчанию во многом схоже с трассировкой UDP соединений. Первому пакету присваивается статус NEW, а всем последующим - статус ESTABLISHED.

При использовании поведения по-умолчанию, для всех пакетов используется одно и то же значение таймаута, которое можно изменить в `/proc/sys/net/ipv4/netfilter/ip_ct_generic_timeout`. По-умолчанию это значение равно 600 секундам, или 10 минутам. В зависимости от типа трафика, это время может меняться, особенно когда соединение устанавливается по спутниковому каналу.

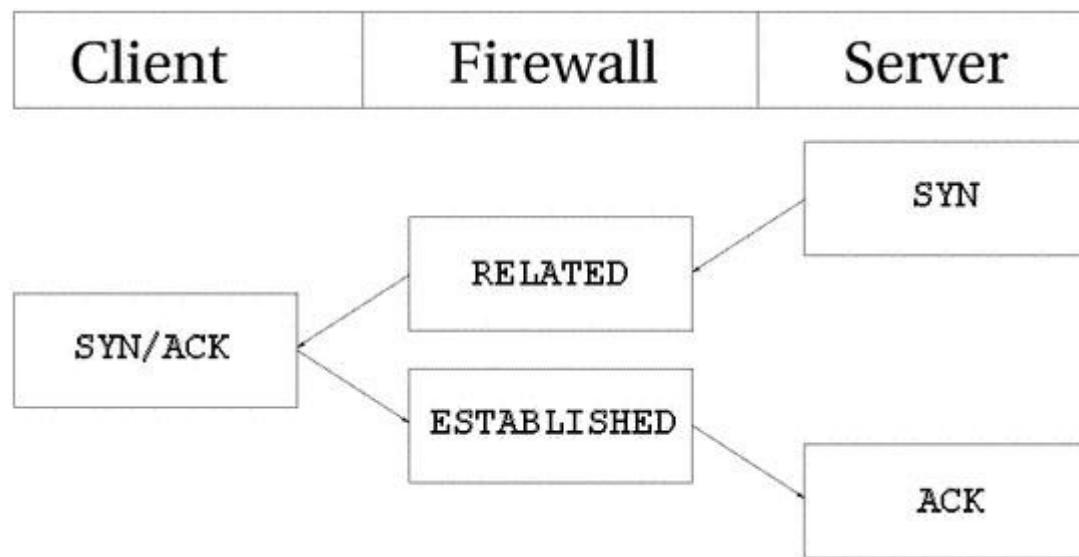
4.8. Трассировка комплексных протоколов

Имеется ряд комплексных протоколов, корректная трассировка которых более сложна. Примером могут служить протоколы ICQ, IRC и FTP. Каждый из этих протоколов несет дополнительную информацию о соединении в области данных пакета. Соответственно корректная трассировка таких соединений требует подключения дополнительных вспомогательных модулей.

В качестве первого примера рассмотрим протокол FTP. Протокол FTP сначала открывает одиночное соединение, которое называется "сеансом управления FTP" (FTP control session). При выполнении команд в пределах этого сеанса, для передачи сопутствующих данных открываются дополнительные порты. Эти соединения могут быть активными или пассивными. При создании активного соединения клиент передает FTP серверу номер порта и IP адрес для соединения. Затем клиент открывает порт, сервер подключает к заданному порту клиента свой порт с номером 20 (известный как FTP-Data) и передает данные через установленное соединение.

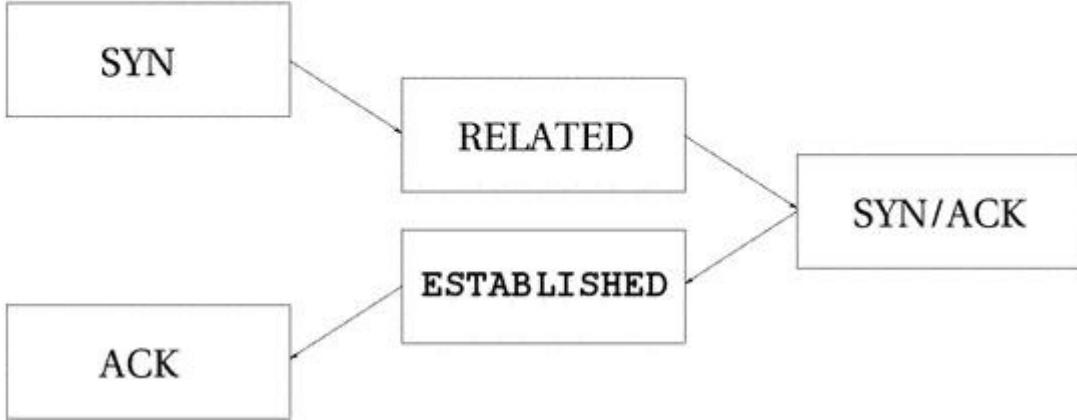
Проблема состоит в том, что брандмауэр ничего не знает об этих дополнительных подключениях, поскольку вся информация о них передается через область данных пакетов. Из-за этого брандмауэр не позволит серверу соединиться с указанным портом клиента.

Решение проблемы состоит в добавлении специального вспомогательного модуля трассировки, который отслеживает, специфичную для данного протокола, информацию в области данных пакетов, передаваемых в рамках сеанса управления. При создании такого соединения, вспомогательный модуль корректно воспримет передаваемую информацию и создаст соответствующую запись в таблице трассировщика со статусом RELATED, благодаря чему соединение будет установлено. Рисунок ниже поясняет порядок выполнения подобного соединения.



Пассивный FTP действует противоположным образом. Клиент посылает запрос серверу на получение данных, а сервер возвращает клиенту IP адрес и номер порта для подключения. Клиент подключает свой 20-й порт (FTP-data) к указанному порту сервера и получает запрошенные данные. Если ваш FTP сервер находится за брандмауэром, то вам потребуется этот вспомогательный модуль для того, чтобы сервер смог обслуживать клиентов из Интернет. То же самое касается случая, когда вы хотите ограничить своих пользователей только возможностью подключения к HTTP и FTP серверам в Интернет и закрыть все остальные порты. Рисунок ниже показывает как выполняется пассивное соединение FTP:

| Client | Firewall | Server |
|--------|----------|--------|
|--------|----------|--------|



Некоторые вспомогательные модули уже включены в состав ядра. Если быть более точным, то в состав ядра включены вспомогательные модули для протоколов **FTP** и **IRC**. Если в вашем распоряжении нет необходимого вспомогательного модуля, то вам следует обратиться к **patch-o-matic**, который содержит большое количество вспомогательных модулей для трассировки таких протоколов, как **ntalk** или **H.323**. Если и здесь вы не нашли то, что вам нужно, то у вас есть еще варианты: вы можете обратиться к **CVS iptables**, если искомый вспомогательный модуль еще не был включен в **patch-o-matic**, либо можете войти в контакт с разработчиками **netfilter** и узнать у них - имеется ли подобный модуль и планируется ли он к выпуску. Если и тут вы потерпели неудачу, то наверное вам следует прочитать [Rusty Russell's Unreliable Netfilter Hacking HOW-TO](#).

Вспомогательные модули могут быть скомпилированы как в виде подгружаемых модулей ядра, так и статически связаны с ядром. Если они скомпилированы как модули, то вы можете загрузить их командой:

```
modprobe ip_conntrack_*
```

Обратите внимание на то, что механизм определения состояния не имеет никакого отношения к трансляции сетевых адресов (NAT), поэтому вам может потребоваться большее количество дополнительных модулей, если вы выполняете такую трансляцию. Допустим, что вы выполняете трансляцию адресов и трассировку FTP соединений, тогда вам необходим так же и соответствующий вспомогательный модуль NAT. Имена вспомогательных модулей NAT начинаются с **ip_nat_**, в соответствии с соглашением об именах. В данном случае модуль называется **ip_nat_ftp**. Для протокола **IRC** такой модуль будет называться **ip_nat_irc**. Тому же самому соглашению следуют и названия вспомогательных модулей трассировщика, например: **ip_conntrack_ftp** и **ip_conntrack_irc**.

Глава 5. Сохранение и восстановление больших наборов правил

В состав пакета **iptables** входят две очень удобные утилиты, особенно если вам приходится иметь дело с большими наборами правил. Называются они **iptables-save** и **iptables-restore**. Первая из них сохраняет, а вторая восстанавливает наборы правил в/из файла. По своему формату файл с набором правил похож на обычные файлы сценариев командной оболочки (**Shell**), в чем вы сможете убедиться чуть ниже.

5.1. Плюсы

Один из плюсов использования утилит **iptables-save** и **iptables-restore** состоит в высокой скорости загрузки и сохранения больших наборов правил. Главный недостаток, связанный с установкой наборов правил из сценариев командной оболочки состоит в том, что команда **iptables** копирует набор правил из пространства ядра в пространство пользователя, вставляет, добавляет или изменяет правило и, наконец, весь набор правил копируется обратно в пространство ядра. Эта последовательность действий выполняется для каждого правила, которое вставляется или изменяется в наборе правил.

Эта проблема легко решается с помощью `iptables-save` и `iptables-restore`. Утилита `iptables-save` записывает набор правил в обычный текстовый файл в особом формате. Утилита `iptables-restore` загружает набор правил из файла. Главное преимущество этих утилит состоит в том, что они производят сохранение/восстановление всего набора правил за одно обращение. `iptables-save` "в один присест" получает из пространства ядра и записывает в файл весь набор правил, а `iptables-restore` загружает из файла и переписывает за одно обращение в пространство ядра набор правил для каждой таблицы. Или другими словами - вместо того, чтобы обращаться огромное число раз к ядру для того чтобы получить набор правил, а затем опять записать его в пространство ядра не меньшее число раз, можно просто сохранить набор правил в файл, а затем загружать его из файла, при этом число перемещений наборов в ядро будет зависеть только от числа используемых таблиц.

Вы уже наверняка поняли, что эти утилиты могут представлять для вас интерес, особенно если вам приходится загружать огромные наборы правил. Однако использование этих утилит имеет и свои отрицательные стороны, которые мы рассмотрим в следующем разделе.

5.2. И минусы

У вас может сложиться впечатление, что `iptables-restore` может обрабатывать своего рода сценарии. Пока не может и вероятнее всего - никогда не сможет. В этом и состоит главный недостаток `iptables-restore`. Чтобы было более понятно - представьте себе случай, когда брандмауэр получает динамический IP-адрес и вы хотите вставить его значение в свои правила во время загрузки системы. Решить эту проблему с помощью `iptables-restore` практически невозможно.

Как одно из решений можно предложить написать небольшой скрипт, который определяет значение IP-адреса и затем вставляет его в набор правил (например, с помощью `sed`) на место некоторого ключевого слова. Здесь вам потребуется создать временный файл, в котором производятся изменения и который затем загружается с помощью `iptables-restore`. Однако такой вариант решения порождает свои проблемы -- вам придется отказаться от утилиты `iptables-save` поскольку она может затереть, созданную вручную, заготовку файла с правилами для `iptables-restore`. В общем - довольно неуклюжее решение.

Еще один вариант - хранить в файле для `iptables-restore` только статические правила, а затем с помощью небольшого скрипта добавлять правила с динамическими параметрами. Конечно же вы уже поняли, что это решение такое же неуклюжее как и первое. Вам придется смириться с тем, что `iptables-restore` не очень хорошо подходит для случая с динамически назначаемым IP-адресом и вообще для случаев, когда вам потребуется динамически изменять набор правил в зависимости от конфигурации системы и т.п..

Еще один недостаток `iptables-restore` и `iptables-save` в том, что их функциональность не всегда соответствует описанной. Проблема состоит в том, что не многие пользуются этими утилитами, еще меньше людей вовлечено в процесс поиска ошибок в этих программах. Поэтому, при использовании некоторых, вновь появившихся, критериев или действий вы можете столкнуться с неожиданным поведением своих правил. Несмотря на возможное существование некоторых проблем, я все же настоятельно рекомендую к использованию эти два инструмента, которые прекрасно работают в большинстве случаев, исключение могут составлять лишь некоторые новые критерии и действия.

5.3. iptables-save

Утилита `iptables-save`, как я уже упоминал, предназначена для сохранения текущего набора правил в файл, который затем может быть использован утилитой `iptables-restore`. Эта команда очень проста в использовании и имеет всего два аргумента.

`iptables-save [-c] [-t table]`

Первый аргумент `-c` (допустимо использовать более длинный вариант `--counters`) заставляет `iptables-save` сохранить значения счетчиков байт и пакетов. Это делает возможным рестарт брандмауэра без потери счетчиков, которые могут использоваться для подсчета статистики. По умолчанию, при запуске без ключа `-c`, сохранение счетчиков не производится.

С помощью ключа `-t` (более длинный вариант `--table`) можно указать имя таблицы для сохранения. Если ключ `-t` не задан, то сохраняются все таблицы. Ниже приведен пример работы команды `iptables-save` в случае, когда набор не содержит ни одного правила.

```
# Generated by iptables-save v1.2.6a on Wed Apr 24 10:19:17 2002
*filter
:INPUT ACCEPT [404:19766]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [530:43376]
COMMIT
# Completed on Wed Apr 24 10:19:17 2002
# Generated by iptables-save v1.2.6a on Wed Apr 24 10:19:17 2002
*mangle
:PREROUTING ACCEPT [451:22060]
:INPUT ACCEPT [451:22060]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [594:47151]
:POSTROUTING ACCEPT [594:47151]
COMMIT
# Completed on Wed Apr 24 10:19:17 2002
# Generated by iptables-save v1.2.6a on Wed Apr 24 10:19:17 2002
*nat
:PREROUTING ACCEPT [0:0]
:POSTROUTING ACCEPT [3:450]
:OUTPUT ACCEPT [3:450]
COMMIT
# Completed on Wed Apr 24 10:19:17 2002
```

Строки, начинающиеся с символа `#`, являются комментариями. Имена таблиц начинаются с символа `*` (звездочка), например: `*mangle`. После каждого имени таблицы следуют описания цепочек и правил. Описания цепочек записываются в формате :

`<chain-name> <chain-policy> [<packet-counter>:<byte-counter>],`,
где `<chain-name>` - это название цепочки (например `PREROUTING`), `<chain-policy>` - политика по-умолчанию (например `ACCEPT`). Завершают описание цепочки значения счетчиков пакетов и байт, те самые счетчики, которые вы получите в результате выполнения команды `iptables -L -v`. Описание каждой таблицы завершает ключевое слово `COMMIT`, которое означает, что в этой точке набор правил для данной таблицы будет передан в пространство ядра.

Пример выше показал как выглядит содержимое пустого набора правил, сохраненного утилитой `iptables-save`. Ниже показан результат сохранения небольшого набора правил (`Iptables-save ruleset`):

```
# Generated by iptables-save v1.2.6a on Wed Apr 24 10:19:55 2002
*filter
:INPUT DROP [1:229]
:FORWARD DROP [0:0]
:OUTPUT DROP [0:0]
[0:0] -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
[0:0] -A FORWARD -i eth0 -m state --state RELATED,ESTABLISHED -j ACCEPT
[0:0] -A FORWARD -i eth1 -m state --state NEW,RELATED,ESTABLISHED -j ACCEPT
[0:0] -A OUTPUT -m state --state NEW,RELATED,ESTABLISHED -j ACCEPT
COMMIT
# Completed on Wed Apr 24 10:19:55 2002
# Generated by iptables-save v1.2.6a on Wed Apr 24 10:19:55 2002
*mangle
```

```

:PREROUTING ACCEPT [658:32445]
:INPUT ACCEPT [658:32445]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [891:68234]
:POSTROUTING ACCEPT [891:68234]
COMMIT
# Completed on Wed Apr 24 10:19:55 2002
# Generated by iptables-save v1.2.6a on Wed Apr 24 10:19:55 2002
*nat
:PREROUTING ACCEPT [1:229]
:POSTROUTING ACCEPT [3:450]
:OUTPUT ACCEPT [3:450]
[0:0] -A POSTROUTING -o eth0 -j SNAT --to-source 195.233.192.1
COMMIT
# Completed on Wed Apr 24 10:19:55 2002

```

Из примера виден результат действия аргумента `-c` - перед каждым правилом и в строке описания каждой цепочки имеются числа, отображающие содержимое счетчиков пакетов и байт. Сразу замечу, что набор правил утилиты `iptables-save` выдает на стандартный вывод, поэтому, при сохранении набора в файл команда должна выглядеть примерно так:

```
iptables-save -c > /etc/iptables-save
```

Эта команда запишет весь набор правил, вместе с содержимым счетчиков, в файл с именем `/etc/iptables-save`.

5.4. `iptables-restore`

Утилита `iptables-restore` используется для восстановления (загрузки) набора правил, который ранее был сохранен утилитой `iptables-save`. Набор правил утилиты получает со стандартного ввода и не может загружать его из файла напрямую. Команда имеет следующий синтаксис:

```
iptables-restore [-c] [-n]
```

Ключ `-c` (более длинный вариант `--counters`) заставляет восстанавливать значения счетчиков.

Указание ключа `-n` (более длинный вариант `--noflush`) сообщает `iptables-restore` о том, что правила должны быть добавлены к имеющимся. По-умолчанию утилиты `iptables-restore` (без ключа `-n`) очистит содержимое таблиц и цепочек перед загрузкой нового набора правил.

Для загрузки набора правил утилитой `iptables-restore` из файла можно предложить несколько вариантов, но наиболее употребимый:

```
cat /etc/iptables-save | iptables-restore -c
```

В результате выполнения этой команды содержимое файла `/etc/iptables-save` будет прочитано утилитой `cat` и перенаправлено на стандартный ввод утилиты `iptables-restore`. Можно было бы привести еще целый ряд команд, с помощью которых можно организовать загрузку набора правил из файла, но это выходит за рамки темы, поэтому оставлю читателю возможность самому найти более удобный для него вариант.

После исполнения этой команды набор правил должен загрузиться и все должно работать. Если это не так, то скорее всего вы допустили ошибку при наборе команды.

Глава 6. Как строить правила

В данной главе будет обсуждаться порядок построения собственных правил для `iptables`. Каждая строка, которую вы вставляете в ту или иную цепочку, должна содержать отдельное правило. Мы так же обсудим основные критерии и действия (`targets`) и порядок создания своих собственных действий (т.е. подцепочек правил).

6.1. Основы

Как уже говорилось выше, каждое правило - это строка, содержащая в себе критерии определяющие, подпадает ли пакет под заданное правило, и действие, которое необходимо выполнить в случае выполнения критерия. В общем виде правила записываются примерно так:

```
iptables [-t table] command [match] [target/jump]
```

Нигде не утверждается, что описание действия (`target/jump`) должно стоять последним в строке, однако, такая нотация более удобочитаема. Как бы то ни было, но чаще всего вам будет встречаться именно такой способ записи правил.

Если в правило не включается спецификатор `[-t table]`, то по умолчанию предполагается использование таблицы `filter`, если же предполагается использование другой таблицы, то это требуется указать явно. Спецификатор таблицы так же можно указывать в любом месте строки правила, однако более или менее стандартом считается указание таблицы в начале правила.

Далее, непосредственно за именем таблицы, должна стоять команда. Если спецификатора таблицы нет, то команда всегда должна стоять первой. Команда определяет действие `iptables`, например: вставить правило, или добавить правило в конец цепочки, или удалить правило и т.п.

Раздел `match` задает критерии проверки, по которым определяется подпадает ли пакет под действие этого правила или нет. Здесь мы можем указать самые разные критерии - IP-адрес источника пакета или сети, IP-адрес места назначения, порт, протокол, сетевой интерфейс и т.д. Существует множество разнообразных критериев, но об этом - несколько позже.

И наконец `target` указывает, какое действие должно быть выполнено при условии выполнения критериев в правиле. Здесь можно заставить ядро передать пакет в другую цепочку правил, "бросить" пакет и забыть про него, выдать на источник сообщение об ошибке и т.п.

6.2. Таблицы

Опция `-t` указывает на используемую таблицу. По умолчанию используется таблица `filter`. С ключом `-t` применяются следующие опции. **Таблица 6-1. Таблицы** Состояние Описание

nat Таблица `nat` используется главным образом для преобразования сетевых адресов (Network Address Translation). Через эту таблицу проходит только первый пакет из потока.

Преобразования адресов автоматически применяется ко всем последующим пакетам. Это один из факторов, исходя из которых мы не должны осуществлять какую-либо фильтрацию в этой таблице. Цепочка `PREROUTING` используется для внесения изменений в пакеты на входе в брандмауэр. Цепочка `OUTPUT` используется для преобразования адресов в пакетах, созданных приложениями внутри брандмауэра, перед принятием решения о маршрутизации. И последняя цепочка в этой таблице -- `POSTROUTING`, которая используется для преобразования пакетов перед выдачей их в сеть.

mangle Эта таблица используется для внесения изменений в заголовки пакетов. Примером может служить изменение поля `TTL`, `TOS` или `MARK`. Важно: в действительности поле `MARK` не изменяется, но в памяти ядра заводится структура, которая сопровождает данный пакет все время его прохождения через брандмауэр, так что другие правила и приложения на данном брандмауэре (и только на данной брандмауэре) могут использовать это поле в своих целях. Таблица имеет пять цепочек `PREROUTING`, `POSTROUTING`, `INPUT`, `OUTPUT` и `FORWARD`. `PREROUTING` используется для внесения изменений на входе в брандмауэр, перед принятием решения о маршрутизации. `POSTROUTING` используется для внесения изменений на выходе из брандмауэра, после принятия решения о маршрутизации. `INPUT` - для внесения изменений в пакеты перед тем как они будут переданы локальному приложению внутри брандмауэра. `OUTPUT` - для внесения изменений в пакеты, поступающие от приложений внутри брандмауэра. `FORWARD` - для внесения изменений в транзитные пакеты после первого принятия решения о проприоритете, но перед последним принятием решения о проприоритете. Замечу, что таблица `mangle` ни в коем случае не должна использоваться для преобразования сетевых адресов или маскарадинга (Network Address Translation, Masquerading), поскольку для этих целей имеется таблица `@nat`.

filter Таблица **filter** используется главным образом для фильтрации пакетов. Для примера, здесь мы можем выполнить **DROP**, **LOG**, **ACCEPT** или **REJECT** без каких либо ограничений, которые имеются в других таблицах. Имеется три встроенных цепочки. Первая - **FORWARD**, используемая для фильтрации пакетов, идущих транзитом через брандмауэр. Цепочку **INPUT** проходят пакеты, которые предназначены локальным приложениям (брандмауэру). И цепочка **OUTPUT** - используется для фильтрации исходящих пакетов, генерированных приложениями на самом брандмауэре.

Выше мы рассмотрели основные отличия трех имеющихся таблиц. Каждая из них должна использоваться только в своих целях, и вы должны это понимать. Нецелевое использование таблиц может привести к ослаблению защиты брандмауэра и сети, находящейся за ним. Позднее, в главе **Порядок прохождения таблиц и цепочек**, мы подробнее остановимся на этом.

6.3. Команды

Ниже приводится список команд и правила их использования. Посредством команд мы сообщаем **iptables**, что мы предполагаем сделать. Обычно предполагается одно из двух действий - добавление нового правила в цепочку или удаление существующего правила из той или иной таблицы. Далее приведены команды, которые используются в **iptables**.

Таблица 6-2. Команды Команда **-A**, **--append**

Пример **iptables -A INPUT ...**

Описание Добавляет новое правило в конец заданной цепочки.

Команда **-D**, **--delete**

Пример **iptables -D INPUT --dport 80 -j DROP**, **iptables -D INPUT 1**

Описание Удаление правила из цепочки. Команда имеет два формата записи, первый - когда задается критерий сравнения с опцией **-D** (см. первый пример), второй - порядковый номер правила. Если задается критерий сравнения, то удаляется правило, которое имеет в себе этот критерий, если задается номер правила, то будет удалено правило с заданным номером. Счет правил в цепочках начинается с 1.

Команда **-R**, **--replace**

Пример **iptables -R INPUT 1 -s 192.168.0.1 -j DROP**

Описание Эта команда заменяет одно правило другим. В основном она используется во время отладки новых правил.

Команда **-I**, **--insert**

Пример **iptables -I INPUT 1 --dport 80 -j ACCEPT**

Описание Вставляет новое правило в цепочку. Число, следующее за именем цепочки указывает номер правила, перед которым нужно вставить новое правило, другими словами число задает номер для вставляемого правила. В примере выше, указывается, что данное правило должно быть 1-м в цепочке **INPUT**.

Команда **-L**, **--list**

Пример **iptables -L INPUT**

Описание Вывод списка правил в заданной цепочке, в данном примере предполагается вывод правил из цепочки **INPUT**. Если имя цепочки не указывается, то выводится список правил для всех цепочек. Формат вывода зависит от наличия дополнительных ключей в команде, например **-n**, **-v**, и пр.

Команда **-F**, **--flush**

Пример **iptables -F INPUT**

Описание Сброс (удаление) всех правил из заданной цепочки (таблицы). Если имя цепочки и таблицы не указывается, то удаляются все правила, во всех цепочках. (Хочется от себя добавить, что если не указана таблица ключом **-t** (**--table**), то очистка цепочек производится только в таблице **filter**, прим. перев.)

Команда -Z, --zero

Пример iptables -Z INPUT

Описание Обнуление всех счетчиков в заданной цепочке. Если имя цепочки не указывается, то подразумеваются все цепочки. При использовании ключа **-v** совместно с командой **-L**, на вывод будут поданы и состояния счетчиков пакетов, попавших под действие каждого правила. Допускается совместное использование команд **-L** и **-Z**. В этом случае будет выдан сначала список правил со счетчиками, а затем произойдет обнуление счетчиков.

Команда -N, --new-chain

Пример iptables -N allowed

Описание Создается новая цепочка с заданным именем в заданной таблице. В выше приведенном примере создается новая цепочка с именем **allowed**. Имя цепочки должно быть уникальным и не должно совпадать с зарезервированными именами цепочек и действий (такими как **DROP**, **REJECT** и т.п.)

Команда -X, --delete-chain

Пример iptables -X allowed

Описание Удаление заданной цепочки из заданной таблицы. Удаляемая цепочка не должна иметь правил и не должно быть ссылок из других цепочек на удаляемую цепочку. Если имя цепочки не указано, то будут удалены все цепочки заданной таблице кроме встроенных.

Команда -P, --policy

Пример iptables -P INPUT DROP

Описание Задает политику по-умолчанию для заданной цепочки. Политика по-умолчанию определяет действие, применяемое к пакетам не попавшим под действие ни одного из правил в цепочке. В качестве политики по умолчанию допускается использовать **DROP** и **ACCEPT**.

Команда -E, --rename-chain

Пример iptables -E allowed disallowed

Описание Команда **-E** выполняет переименование пользовательской цепочки. В примере цепочка **allowed** будет переименована в цепочку **disallowed**. Эти переименования не изменяют порядок работы, а носят только косметический характер.

Команда должна быть указана всегда. Список доступных команд можно просмотреть с помощью команды **iptables -h** или, что тоже самое, **iptables --help**. Некоторые команды могут использоваться совместно с дополнительными ключами. Ниже приводится список дополнительных ключей и описывается результат их действия. При этом заметьте, что здесь не приводится дополнительных ключей, которые используются при построении критериев (**matches**) или действий (**targets**). Эти опции мы будем обсуждать далее.

Таблица 6-3. Дополнительные ключи Ключ -v, --verbose

Команды, с которыми используется **--list**, **--append**, **--insert**, **--delete**, **--replace**

Описание Используется для повышения информативности вывода и, как правило, используется совместно с командой **--list**. В случае использования с командой **--list**, в вывод этой команды включаются так же имя интерфейса, счетчики пакетов и байт для каждого правила. Формат вывода счетчиков предполагает вывод кроме цифр числа еще и символьные множители K (x1000), M (x1,000,000) и G (x1,000,000,000). Для того, чтобы заставить команду **--list** выводить полное число (без употребления множителей) требуется применять ключ **-x**, который описан ниже. Если ключ **-v**, **--verbose** используется с командами **--append**, **--insert**, **--delete** или **--replace**, то будет выведен подробный отчет о произведенной операции.

Ключ -x, --exact

Команды, с которыми используется --list

Описание Для всех чисел в выходных данных выводятся их точные значения без округления и без использования множителей K, M, G. Этот ключ используется только с командой `--list` и не применим с другими командами.

Ключ -n, --numeric

Команды, с которыми используется --list

Описание Заставляет `iptables` выводить IP-адреса и номера портов в числовом виде предотвращая попытки преобразовать их в символические имена. Данный ключ используется только с командой `--list`.

Ключ --line-numbers

Команды, с которыми используется --list

Описание Ключ `--line-numbers` включает режим вывода номеров строк при отображении списка правил командой `--list`. Номер строки соответствует позиции правила в цепочке. Этот ключ используется только с командой `--list`.

Ключ -c, --set-counters

Команды, с которыми используется --insert, --append, --replace

Описание Этот ключ используется для установки начального значения счетчиков пакетов и байт в заданное значение при создании нового правила. Например, ключ `--set-counters 20 4000` установит счетчик пакетов = 20, а счетчик байт = 4000.

Ключ --modprobe

Команды, с которыми используется Все

Описание Ключ `--modprobe` определяет команду загрузки модуля ядра. Данный ключ может использоваться в случае, когда модули ядра находятся вне пути поиска (`search path`). Этот ключ может использоваться с любой командой.

6.4. Критерии

Здесь мы подробнее остановимся на критериях выделения пакетов. Я разбил все критерии на пять групп. Первая - общие критерии которые могут использоваться в любых правилах. Вторая - TCP критерии которые применяются только к TCP пакетам. Третья - UDP критерии которые применяются только к UDP пакетам. Четвертая - ICMP критерии для работы с ICMP пакетами. И наконец пятая - специальные критерии, такие как `state`, `owner`, `limit` и пр.

6.4.1. Общие критерии

Здесь мы рассмотрим Общие критерии. Общие критерии допустимо употреблять в любых правилах, они не зависят от типа протокола и не требуют подгрузки модулей расширения. К этой группе я умышленно отнес критерий `--protocol` несмотря на то, что он используется в некоторых специфичных от протокола расширениях. Например, мы решили использовать TCP критерий, тогда нам необходимо будет использовать и критерий `--protocol` которому в качестве дополнительного ключа передается название протокола - TCP. Однако критерий `--protocol` сам по себе является критерием, который используется для указания типа протокола.

Таблица 6-4. Общие критерии Критерий -p, --protocol

Пример `iptables -A INPUT -p tcp`

Описание Этот критерий используется для указания типа протокола. Примерами протоколов могут быть TCP, UDP и ICMP. Список протоколов можно посмотреть в файле /etc/protocols. Прежде всего, в качестве имени протокола в данный критерий можно передавать один из трех вышеупомянутых протоколов, а также ключевое слово ALL. В качестве протокола допускается передавать число - номер протокола, так например, протоколу ICMP соответствует число 1, TCP - 6 и UDP - 17. Соответствия между номерами протоколов и их именами вы можете посмотреть в файле /etc/protocols, который уже упоминался. Критерию может передаваться и список протоколов, разделенных запятыми, например так: udp, tcp (Хотя автор и указывает на возможность передачи списка протоколов, тем не менее вам вряд ли удастся это сделать! Кстати, man iptables явно оговаривает, что в данном критерии может быть указан только один протокол. Может быть это расширение имеется в patch-o-matic? прим. перев.) Если данному критерию передается числовое значение 0, то это эквивалентно использованию спецификатора ALL, который подразумевается по умолчанию, когда критерий --protocol не используется. Для логической инверсии критерия, перед именем протокола (списком протоколов) используется символ !, например --protocol ! tcp подразумевает пакеты протоколов, UDP и ICMP.

Критерий -s, --src, --source

Пример iptables -A INPUT -s 192.168.1.1

Описание IP-адрес(а) источника пакета. Адрес источника может указываться так, как показано в примере, тогда подразумевается единственный IP-адрес. А можно указать адрес в виде address/mask, например как 192.168.0.0/255.255.255.0, или более современным способом 192.168.0.0/24, т.е. фактически определяя диапазон адресов Как и ранее, символ !, установленный перед адресом, означает логическое отрицание, т.е. --source ! 192.168.0.0/24 означает любой адрес кроме адресов 192.168.0.x .

Критерий -d, --dst, --destination

Пример iptables -A INPUT -d 192.168.1.1

Описание IP-адрес(а) получателя. Имеет синтаксис схожий с критерием --source, за исключением того, что подразумевает адрес места назначения. Точно так же может определять как единственный IP-адрес, так и диапазон адресов. Символ ! используется для логической инверсии критерия.

Критерий -i, --in-interface

Пример iptables -A INPUT -i eth0

Описание Интерфейс, с которого был получен пакет. Использование этого критерия допускается только в цепочках INPUT, FORWARD и PREROUTING, в любых других случаях будет вызывать сообщение об ошибке. При отсутствии этого критерия предполагается любой интерфейс, что равносильно использованию критерия -i +. Как и прежде, символ ! инвертирует результат совпадения. Если имя интерфейса завершается символом +, то критерий задает все интерфейсы, начинающиеся с заданной строки, например -i PPP+ обозначает любой PPP интерфейс, а запись -i ! eth+ - любой интерфейс, кроме любого eth.

Критерий -o, --out-interface

Пример iptables -A FORWARD -o eth0

Описание Задает имя выходного интерфейса. Этот критерий допускается использовать только в цепочках OUTPUT, FORWARD и POSTROUTING, в противном случае будет генерироваться сообщение об ошибке. При отсутствии этого критерия предполагается любой интерфейс, что равносильно использованию критерия -o +. Как и прежде, символ ! инвертирует результат совпадения. Если имя интерфейса завершается символом +, то критерий задает все интерфейсы, начинающиеся с заданной строки, например -o eth+ обозначает любой eth интерфейс, а запись -o ! eth+ - любой интерфейс, кроме любого eth.

Критерий -f, --fragment

Пример iptables -A INPUT -f

Описание Правило распространяется на все фрагменты фрагментированного пакета, кроме первого, сделано это потому, что нет возможности определить исходящий/входящий порт для фрагмента пакета, а для ICMP-пакетов определить их тип. С помощью фрагментированных пакетов могут производиться атаки на ваш брандмауэр, так как фрагменты пакетов могут не отлавливаться другими правилами. Как и раньше, допускается использования символа ! для инверсии результата сравнения. только в данном случае символ ! должен предшествовать критерию -f, например ! -f. Инверсия критерия трактуется как "все первые фрагменты фрагментированных пакетов и/или нефрагментированные пакеты, но не вторые и последующие фрагменты фрагментированных пакетов".

6.4.2. Неявные критерии

В этом разделе мы рассмотрим неявные критерии, точнее, те критерии, которые подгружаются неявно и становятся доступны, например при указании критерия --protocol tcp. На сегодняшний день существует три автоматически подгружаемых расширения, это TCP критерии, UDP критерии и ICMP критерии (при построении своих правил я столкнулся с необходимостью явного указания ключа -m tcp, т.е. о неявности здесь говорить не приходится, поэтому будьте внимательнее при построении своих правил, если что-то не идет - пробуйте явно указывать необходимое расширение. прим. перев.). Загрузка этих расширений может производиться и явным образом с помощью ключа -m, -match, например -m tcp.

6.4.2.1. TCP критерии

Этот набор критериев зависит от типа протокола и работает только с TCP пакетами. Чтобы использовать их, вам потребуется в правилах указывать тип протокола --protocol tcp. Важно: критерий --protocol tcp обязательно должен стоять перед специфичным критерием. Эти расширения загружаются автоматически как для tcp протокола, так и для udp и icmp протоколов. (О неявной загрузке расширений я уже упоминал выше прим. перев.).

Таблица 6-5. TCP критерии Критерий --sport, --source-port

Пример iptables -A INPUT -p tcp --sport 22

Описание Исходный порт, с которого был отправлен пакет. В качестве параметра может указываться номер порта или название сетевой службы. Соответствие имен сервисов и номеров портов вы сможете найти в файле /etc/services. При указании номеров портов правила отрабатывают несколько быстрее. однако это менее удобно при разборе листингов скриптов. Если же вы собираетесь создавать значительные по объему наборы правил, скажем порядка нескольких сотен и более, то тут предпочтительнее использовать номера портов. Номера портов могут задаваться в виде интервала из минимального и максимального номеров, например --source-port 22:80. Если опускается минимальный порт, т.е. когда критерий записывается как --source-port :80, то в качестве начала диапазона принимается число 0. Если опускается максимальный порт, т.е. когда критерий записывается как --source-port 22:, то в качестве конца диапазона принимается число 65535. Допускается такая запись --source-port 80:22, в этом случае iptables поменяет числа 22 и 80 местами, т.е. подобного рода запись будет преобразована в --source-port 22:80. Как и раньше, символ ! используется для инверсии. Так критерий --source-port ! 22 подразумевает любой порт, кроме 22. Инверсия может применяться и к диапазону портов, например --source-port ! 22:80. За дополнительной информацией обращайтесь к описанию критерия multiport.

Критерий --dport, --destination-port

Пример iptables -A INPUT -p tcp --dport 22

Описание Порт или диапазон портов, на который адресован пакет. Аргументы задаются в том же формате, что и для --source-port.

Критерий --tcp-flags

Пример iptables -p tcp --tcp-flags SYN,FIN,ACK SYN

Описание Определяет маску и флаги tcp-пакета. Пакет считается удовлетворяющим критерию, если из перечисленных флагов в первом списке в единичное состояние установлены флаги из второго списка. Так для вышеуказанного примера под критерий подпадают пакеты у которых флаг SYN установлен, а флаги FIN и ACK сброшены. В качестве аргументов критерия могут выступать флаги SYN, ACK, FIN, RST, URG, PSH, а так же зарезервированные идентификаторы ALL и NONE. ALL - значит ВСЕ флаги и NONE - НИ ОДИН флаг. Так, критерий `--tcp-flags ALL NONE` означает - "все флаги в пакете должны быть сброшены". Как и ранее, символ ! означает инверсию критерия Важно: имена флагов в каждом списке должны разделяться запятыми, пробелы служат для разделения списков.

Критерий `--syn`

Пример `iptables -p tcp --syn`

Описание Критерий `--syn` является по сути реликтом, перекочевавшим из `ipchains`. Критерию соответствуют пакеты с установленным флагом SYN и сброшенными флагами ACK и FIN. Этот критерий аналогичен критерию `--tcp-flags SYN,ACK,FIN SYN`. Такие пакеты используются для открытия соединения TCP. Заблокировав такие пакеты, вы надежно заблокируете все входящие запросы на соединение, однако этот критерий не способен заблокировать исходящие запросы на соединение. Как и ранее, допускается инвертирование критерия символом !. Так критерий `! --syn` означает - "все пакеты, не являющиеся запросом на соединение", т.е. все пакеты с установленными флагами FIN или ACK.

Критерий `--tcp-option`

Пример `iptables -p tcp --tcp-option 16`

Описание Удовлетворяющим условию данного критерия будет считаться пакет, TCP параметр которого равен заданному числу. TCP Option - это часть заголовка пакета. Она состоит из 3 различных полей. Первое 8-ми битовое поле содержит информацию об опциях, используемых в данном соединении. Второе 8-ми битовое поле содержит длину поля опций. Если следовать стандартам до конца, то следовало бы реализовать обработку всех возможных вариантов, однако, вместо этого мы можем проверить первое поле и в случае, если там указана неподдерживаемая нашим брандмауэром опция, то просто перешагнуть через третье поле (длина которого содержится во втором поле). Пакет, который не будет иметь полного TCP заголовка, будет сброшен автоматически при попытке изучения его TCP параметра. Как и ранее, допускается использование флага инверсии условия !. Дополнительную информацию по TCP Options вы сможете найти на [Internet Engineering Task Force](#)

6.4.2.2. UDP критерии

В данном разделе будут рассматриваться критерии, специфичные только для протокола UDP. Эти расширения подгружаются автоматически при указании типа протокола `--protocol udp`. Важно отметить, что пакеты UDP не ориентированы на установленное соединение, и поэтому не имеют различных флагов которые дают возможность судить о предназначении датаграмм. Получение UDP пакетов не требует какого либо подтверждения со стороны получателя. Если они потеряны, то они просто потеряны (не вызывая передачу ICMP сообщения об ошибке). Это предполагает наличие значительно меньшего числа дополнительных критериев, в отличие от TCP пакетов. Важно: Хороший брандмаэр должен работать с пакетами любого типа, UDP или ICMP, которые считаются не ориентированными на соединение, так же хорошо как и с TCP пакетами. Об этом мы поговорим позднее, в следующих главах.

Таблица 6-6. UDP критерии Критерий `--sport`, `--source-port`

Пример `iptables -A INPUT -p udp --sport 53`

Описание Исходный порт, с которого был отправлен пакет. В качестве параметра может указываться номер порта или название сетевой службы. Соответствие имен сервисов и номеров портов вы сможете найти в файле [services.txt](#). При указании номеров портов правила отрабатывают несколько быстрее. Однако это менее удобно при разборе листингов скриптов. Если же вы собираетесь создавать значительные по объему наборы правил, скажем порядка нескольких сотен и более, то тут предпочтительнее использовать номера портов. Номера портов могут задаваться в виде интервала из

минимального и максимального номеров, например `--source-port 22:80`. Если опускается минимальный порт, т.е. когда критерий записывается как `--source-port :80`, то в качестве начала диапазона принимается число 0. Если опускается максимальный порт, т.е. когда критерий записывается как `--source-port 22: :`, то в качестве конца диапазона принимается число 65535. Допускается такая запись `--source-port 80:22`, в этом случае `iptables` поменяет числа 22 и 80 местами, т.е. подобного рода запись будет преобразована в `--source-port 22:80`. Как и раньше, символ `!` используется для инверсии. Так критерий `--source-port ! 22` подразумевает любой порт, кроме 22. Инверсия может применяться и к диапазону портов, например `--source-port ! 22:80`.

Критерий `--dport`, `--destination-port`

Пример `iptables -A INPUT -p udp --dport 53`

Описание Порт, на который адресован пакет. Формат аргументов полностью аналогичен принятому в критерии `--source-port`.

6.4.2.3. ICMP критерии

Этот протокол используется, как правило, для передачи сообщений об ошибках и для управления соединением. Он не является подчиненным IP протоколу, но тесно с ним взаимодействует, поскольку помогает обрабатывать ошибочные ситуации. Заголовки ICMP пакетов очень похожи на IP заголовки, но имеют и отличия. Главное свойство этого протокола заключается в типе заголовка, который содержит информацию о том, что это за пакет. Например, когда мы пытаемся соединиться с недоступным хостом, то мы получим в ответ сообщение ICMP `host unreachable`. Полный список типов ICMP сообщений, вы можете посмотреть в приложении *Типы ICMP*. Существует только один специфичный критерий для ICMP пакетов. Это расширение загружается автоматически, когда мы указываем критерий `--protocol icmp`. Заметьте, что для проверки ICMP пакетов могут употребляться и общие критерии, поскольку известны и адрес источника и адрес назначения и пр.

Таблица 6-7. ICMP критерии Критерий `--icmp-type`

Пример `iptables -A INPUT -p icmp --icmp-type 8`

Описание Тип сообщения ICMP определяется номером или именем. Числовые значения определяются в RFC 792. Чтобы получить список имен ICMP значений выполните команду `iptables --protocol icmp --help`, или посмотрите приложение *Типы ICMP*. Как и ранее, символ `!` инвертирует критерий, например `--icmp-type ! 8`.

6.4.3.1. Критерий Limit

Должен подгружаться явно ключом `-m limit`. Прекрасно подходит для правил, производящих запись в системный журнал (*logging*) и т.п. Добавляя этот критерий, мы тем самым устанавливаем предельное число пакетов в единицу времени, которое способно пропустить правило. Можно использовать символ `!` для инверсии, например `-m limit ! --limit 5/s`. В этом случае подразумевается, что пакеты будут проходить правило только после превышения ограничения.

Более наглядно этот критерий можно представить себе как некоторую емкость с выпускным отверстием, через которое проходит определенное число пакетов за единицу времени (т.е. скорость "вытекания"). Скорость "вытекания" как раз и определяет величина `--limit`. Величина `--limit-burst` задает общий "объем емкости". А теперь представим себе правило `--limit 3/minute --limit-burst 5`, тогда после поступления 5 пакетов (за очень короткий промежуток времени), емкость "наполнится" и каждый последующий пакет будет вызывать "переполнение" емкости, т.е. "срабатывание" критерия. Через 20 секунд "уровень" в емкости будет понижен (в соответствии с величиной `--limit`), таким образом она готова будет принять еще один пакет, не вызывая "переполнения" емкости, т.е. срабатывания критерия.

Рассмотрим еще подробнее

- Предположим наличие правила, содержащего критерий `-m limit --limit 5/second --limit-burst 10`. Ключ `limit-burst` установил объем "емкости" равный 10-ти. Каждый пакет, который подпадает под указанное правило, направляется в эту емкость.

2. Допустим, в течение 1/1000 секунды, мы получили 10 пакетов, тогда с получением каждого пакета "уровень" в "емкости" будет возрастать: 1-2-3-4-5-6-7-8-9-10.
3. Емкость наполнилась. Теперь пакеты, подпадающие под наше ограничительное правило, больше не смогут попасть в эту "емкость" (там просто нет места), поэтому они (пакеты) пойдут дальше по набору правил, пока не будут явно восприняты одним из них, либо подвергнутся политике по-умолчанию.
4. Каждые 1/5 секунды "уровень" в воображаемой емкости снижается на 1, и так до тех пор, пока "емкость" не будет опустошена. Через секунду, после приема 10-ти пакетов "емкость" готова будет принять еще 5 пакетов.
5. Само собой разумеется, что "уровень" в "емкости" возрастает на 1 с каждым вновь пришедшим пакетом.

От переводчика: Очень долгое время мое понимание критериев `limit` находилось на интуитивном уровне, пока [Владимир Холманов](#) (снимаю шляпу в глубочайшем поклоне) не объяснил мне просто и понятно его суть. Постараюсь передать его пояснения:

1. Расширение `-m limit` подразумевает наличие ключей `--limit` и `--limit-burst`. Если вы не указываете эти ключи, то они принимают значение по-умолчанию.
2. Ключ `--limit-burst` - это максимальное значение счетчика пакетов, при котором срабатывает ограничение.
3. Ключ `--limit` - это скорость, с которой счетчик `burst limit` "откручивается назад".

Принцип, который просто реализуется на С и широко используется во многих алгоритмах-ограничителях.

Таблица 6-8. Ключи критерия `limit` Ключ `--limit`

Пример `iptables -A INPUT -m limit --limit 3/hour`

Описание Устанавливается средняя скорость "освобождения емкости" за единицу времени. В качестве аргумента указывается число пакетов и время. Допустимыми считаются следующие единицы измерения времени: `/second` `/minute` `/hour` `/day`. По умолчанию принято значение 3 пакета в час, или `3/hour`. Использование флага инверсии условия `!` в данном критерии недопустимо.

Ключ `--limit-burst`

Пример `iptables -A INPUT -m limit --limit-burst 5`

Описание Устанавливает максимальное значение числа `burst limit` для критерия `limit`. Это число увеличивается на единицу если получен пакет, подпадающий под действие данного правила, и при этом средняя скорость (задаваемая ключом `--limit`) поступления пакетов уже достигнута. Так происходит до тех пор, пока число `burst limit` не достигнет максимального значения, установленного ключом `--limit-burst`. После этого правило начинает пропускать пакеты со скоростью, задаваемой ключом `--limit`. Значение по-умолчанию принимается равным 5. Для демонстрации принципов работы данного критерия я написал сценарий [limit-match.txt](#) С помощью этого сценария вы увидите как работает критерий `limit`, просто посыпая `ping`-пакеты с различными временными интервалами.

6.4.3.2. Критерий МАС

МАС (Ethernet Media Access Control) критерий используется для проверки исходного МАС-адреса пакета. Расширение `-m mac`, на сегодняшний день, предоставляет единственный критерий, но возможно в будущем он будет расширен и станет более полезен.

Модуль расширения должен подгружаться явно ключом `-m mac`. Упоминаю я об этом потому, что многие, забыв указать этот ключ, удивляются, почему не работает этот критерий.

Таблица 6-9. Ключи критерия МАС Ключ `--mac-source`

Пример `iptables -A INPUT -m mac --mac-source 00:00:00:00:00:01`

Описание MAC адрес сетевого узла, передавшего пакет. MAC адрес должен указываться в форме XX:XX:XX:XX:XX:XX. Как и ранее, символ ! используется для инверсии критерия, например --mac-source ! 00:00:00:00:00:01, что означает - "пакет с любого узла, кроме узла, который имеет MAC адрес 00:00:00:00:00:01". Этот критерий имеет смысл только в цепочках PREROUTING, FORWARD и INPUT и нигде более.

6.4.3.3. Критерий Mark

Критерий mark предоставляет возможность "пометить" пакеты специальным образом. Mark - специальное поле, которое существует только в области памяти ядра и связано с конкретным пакетом. Может использоваться в самых разнообразных целях, например, ограничение трафика и фильтрация. На сегодняшний день существует единственная возможность установки метки на пакет в Linux - это использование действия MARK. Поле mark представляет собой беззнаковое целое число в диапазоне от 0 до 4294967296 для 32-битных систем. **Таблица 6-10. Ключи критерия Mark** Ключ --mark

Пример iptables -t mangle -A INPUT -m mark --mark 1

Описание Критерий производит проверку пакетов, которые были предварительно "помечены". Метки устанавливаются действием MARK, которое мы будем рассматривать ниже. Все пакеты, проходящие через netfilter имеют специальное поле mark. Запомните, что нет никакой возможности передать состояние этого поля вместе с пакетом в сеть. Поле mark является целым беззнаковым, таким образом можно создать не более 4294967296 различных меток. Допускается использовать маску с метками. В данном случае критерий будет выглядеть подобным образом: --mark 1/1. Если указывается маска, то выполняется логическое AND метки и маски.

6.4.3.4. Критерий Multiport

Расширение multiport позволяет указывать в тексте правила несколько портов и диапазонов портов.

Вы не сможете использовать стандартную проверку портов и расширение -m multiport (например --sport 1024:63353 -m multiport --dport 21,23,80) одновременно. Подобные правила будут просто отвергаться iptables.

Таблица 6-11. Ключи критерия Multiport Ключ --source-port

Пример iptables -A INPUT -p tcp -m multiport --source-port 22,53,80,110

Описание Служит для указания списка исходящих портов. С помощью данного критерия можно указать до 15 различных портов. Названия портов в списке должны отделяться друг от друга запятыми, пробелы в списке не допустимы. Данное расширение может использоваться только совместно с критериями -p tcp или -p udp. Главным образом используется как расширенная версия обычного критерия --source-port.

Ключ --destination-port

Пример iptables -A INPUT -p tcp -m multiport --destination-port 22,53,80,110

Описание Служит для указания списка входных портов. Формат задания аргументов полностью аналогичен -m multiport --source-port.

Ключ --port

Пример iptables -A INPUT -p tcp -m multiport --port 22,53,80,110

Описание Данный критерий проверяет как исходящий так и входящий порт пакета. Формат аргументов аналогичен критерию --source-port и --destination-port. Обратите внимание на то что данный критерий проверяет порты обеих направлений, т.е. если вы пишете -m multiport --port 80, то под данный критерий подпадают пакеты, идущие с порта 80 на порт 80.

6.4.3.5. Критерий Owner

Расширение owner предназначено для проверки "владельца" пакета. Изначально данное расширение было написано как пример демонстрации возможностей `iptables`. Допускается использовать этот критерий только в цепочке OUTPUT. Такое ограничение наложено потому, что на сегодняшний день нет реального механизма передачи информации о "владельце" по сети. Справедливости ради следует отметить, что для некоторых пакетов невозможно определить "владельца" в этой цепочке. К такого рода пакетам относятся различные ICMP responses. Поэтому не следует применять этот критерий к ICMP responses пакетам.

Таблица 6-12. Ключи критерия Owner Ключ - -uid-owner

Пример `iptables -A OUTPUT -m owner --uid-owner 500`

Описание Производится проверка "владельца" по User ID (UID). Подобного рода проверка может использоваться, к примеру, для блокировки выхода в Интернет отдельных пользователей.

Ключ - -gid-owner

Пример `iptables -A OUTPUT -m owner --gid-owner 0`

Описание Производится проверка "владельца" пакета по Group ID (GID).

Ключ - -pid-owner

Пример `iptables -A OUTPUT -m owner --pid-owner 78`

Описание Производится проверка "владельца" пакета по Process ID (PID). Этот критерий достаточно сложен в использовании, например, если мы хотим позволить передачу пакетов на HTTP порт только от заданного демона, то нам потребуется написать небольшой сценарий, который получает PID процесса (хотя бы через `ps`) и затем подставляет найденный PID в правила. Пример использования критерия можно найти в [pid-owner.txt](#).

Ключ - -sid-owner

Пример `iptables -A OUTPUT -m owner --sid-owner 100`

Описание Производится проверка Session ID пакета. Значение SID наследуются дочерними процессами от "родителя", так, например, все процессы HTTDPD имеют один и тот же SID (примером таких процессов могут служить HTTDPD Apache и Roxen). Пример использования этого критерия можно найти в [sid-owner.txt](#). Этот сценарий можно запускать по времени для проверки наличия процесса HTTDPD, и в случае отсутствия - перезапустить "упавший" процесс, после чего сбросить содержимое цепочки OUTPUT и ввести ее снова.

6.4.3.6. Критерий State

Критерий state используется совместно с кодом трассировки соединений и позволяет нам получать информацию о признаке состояния соединения, что позволяет судить о состоянии соединения, причем даже для таких протоколов как ICMP и UDP. Данное расширение необходимо загружать явно, с помощью ключа -m state. Более подробно механизм определения состояния соединения обсуждается в разделе *Механизм определения состояний*.

Таблица 6-13. Ключи критерия State Ключ --state

Пример `iptables -A INPUT -m state --state RELATED,ESTABLISHED`

Описание Проверяется признак состояния соединения (state) На сегодняшний день можно указывать 4 состояния: INVALID, ESTABLISHED, NEW и RELATED. INVALID подразумевает, что пакет связан с неизвестным потоком или соединением и, возможно содержит ошибку в данных или в заголовке. Состояние ESTABLISHED указывает на то, что пакет принадлежит уже установленному соединению через которое пакеты идут в обеих направлениях. Признак NEW подразумевает, что пакет открывает новое соединение или пакет принадлежит одностороннему потоку. И наконец, признак RELATED указывает на то что пакет принадлежит уже существующему соединению, но при этом он открывает новое соединение Примером тому может служить передача данных по FTP, или выдача сообщения

ICMP об ошибке, которое связано с существующим TCP или UDP соединением. Замечу, что признак NEW это не то же самое, что установленный бит SYN в пакетах TCP, посредством которых открывается новое соединение, и, подобного рода пакеты, могут быть потенциально опасны в случае, когда для защиты сети вы используете один сетевой экран. Более подробно эта проблема рассматривается ниже в главе *Механизм определения состояний*.

6.4.3.7. Критерий TOS

Критерий TOS предназначен для проведения проверки битов поля TOS. TOS - Type Of Service - представляет собой 8-ми битовое, поле в заголовке IP-пакета. Модуль должен загружаться явно, ключом `-m tos`.

От переводчика: Далее приводится описание поля TOS, взятое не из оригинала, поскольку оригинальное описание я нахожу несколько туманным.

Данное поле служит для нужд маршрутизации пакета. Установка любого бита может привести к тому, что пакет будет обработан маршрутизатором не так как пакет со сброшенными битами TOS. Каждый бит поля TOS имеет свое значение. В пакете может быть установлен только один из битов этого поля, поэтому комбинации не допустимы. Каждый бит определяет тип сетевой службы:

Минимальная задержка Используется в ситуациях, когда время передачи пакета должно быть минимальным, т.е., если есть возможность, то маршрутизатор для такого пакета будет выбирать более скоростной канал. Например, если есть выбор между оптоволоконной линией и спутниковым каналом, то предпочтение будет отдано более скоростному оптоволокну.

Максимальная пропускная способность Указывает, что пакет должен быть переправлен через канал с максимальной пропускной способностью. Например спутниковые каналы, обладая большей задержкой имеют высокую пропускную способность.

Максимальная надежность Выбирается максимально надежный маршрут во избежание необходимости повторной передачи пакета. Примером могут служить PPP и SLIP соединения, которые по своей надежности уступают, к примеру, сетям X.25, поэтому, сетевой провайдер может предусмотреть специальный маршрут с повышенной надежностью.

Минимальные затраты Применяется в случаях, когда важно минимизировать затраты (в смысле деньги) на передачу данных. Например, при передаче через океан (на другой континент) аренда спутникового канала может оказаться дешевле, чем аренда оптоволоконного кабеля. Установка данного бита вполне может привести к тому, что пакет пойдет по более "дешевому" маршруту.

Обычный сервис В данной ситуации все биты поля TOS сброшены. Маршрутизация такого пакета полностью отдается на усмотрение провайдера.

Таблица 6-14. Ключи критерия TOS Ключ --tos

Пример iptables -A INPUT -p tcp -m tos --tos 0x16

Описание Данный критерий предназначен для проверки установленных битов TOS, которые описывались выше. Как правило поле используется для нужд маршрутизации, но вполне может быть использовано с целью "маркировки" пакетов для использования с `iproute2` и дополнительной маршрутизации в linux. В качестве аргумента критерию может быть передано десятичное или шестнадцатиричное число, или мнемоническое описание бита, мнемоники и их числовое значение вы можете получить выполнив команду `iptables -m tos -h`. Ниже приводятся мнемоники и их значения.

Minimize-Delay 16 (0x10) (Минимальная задержка), Maximize-Throughput 8 (0x08) (Максимальная пропускная способность), Maximize-Reliability 4 (0x04) (Максимальная надежность), Minimize-Cost 2 (0x02) (Минимальные затраты), Normal-Service 0 (0x00) (Обычный сервис)

6.4.3.8. Критерий TTL

TTL (Time To Live) является числовым полем в IP заголовке. При прохождении очередного маршрутизатора, это число уменьшается на 1. Если число становится равным нулю, то отправителю пакета будет передано ICMP сообщение типа 11 с кодом 0 (TTL equals 0 during transit) или с кодом 1 (TTL equals 0 during reassembly). Для использования этого критерия необходимо явно загружать модуль ключом -m ttl.

От переводчика: Опять обнаружилось некоторое несоответствие оригинального текста с действительностью, по крайней мере для iptables 1.2.6a, о которой собственно и идет речь, существует три различных критерия проверки поля TTL, это -m ttl --ttl-eq число, -m ttl --ttl-lt число и -m ttl --ttl-gt число. Назначение этих критериев понятно уже из их синтаксиса. Тем не менее, я все таки приведу перевод оригинала: **Таблица 6-15. Ключи критерия TTL**

Ключ --ttl

Пример iptables -A OUTPUT -m ttl --ttl 60

Описание Производит проверку поля TTL на равенство заданному значению. Данный критерий может быть использован при наладке локальной сети, например: для случаев, когда какая либо машина локальной сети не может подключиться к серверу в Интернете, или для поиска "троянов" и пр. Вобщем, области применения этого поля ограничиваются только вашей фантазией. Еще один пример: использование этого критерия может быть направлено на поиск машин с некачественной реализацией стека TCP/IP или с ошибками в конфигурации ОС.

6.4.4. Критерий "мусора" (Unclean match)

Критерий unclean не имеет дополнительных ключей и для его использования достаточно явно загрузить модуль. Будьте осторожны, данный модуль находится еще на стадии разработки и поэтому в некоторых ситуациях может работать некорректно. Данная проверка производится для вычисления пакетов, которые имеют расхождения с принятыми стандартами, это могут быть пакеты с поврежденным заголовком или с неверной контрольной суммой и пр., однако использование этой проверки может привести к разрыву и вполне корректного соединения.

6.5. Действия и переходы

Действия и переходы сообщают правилу, что необходимо выполнить, если пакет соответствует заданному критерию. Чаще всего употребляются действия ACCEPT и DROP. Однако, давайте кратко рассмотрим понятие переходов.

Описание переходов в правилах выглядит точно так же как и описание действий, т.е. ставится ключ -j и указывается название цепочки правил, на которую выполняется переход. На переходы накладывается ряд ограничений, первое - цепочка, на которую выполняется переход, должна находиться в той же таблице, что и цепочка, из которой этот переход выполняется, второе - цепочка , являющаяся целью перехода должна быть создана до того как на нее будут выполняться переходы. Например, создадим цепочку tcp_packets в таблице filter с помощью команды

```
iptables -N tcp_packets
```

Теперь мы можем выполнять переходы на эту цепочку подобно:

```
iptables -A INPUT -p tcp -j tcp_packets
```

Т.е. встретив пакет протокола tcp, iptables произведет переход на цепочку `tcp_packets` и продолжит движение пакета по этой цепочке. Если пакет достиг конца цепочки то он будет возвращен в вызывающую цепочку (в нашем случае это цепочка INPUT) и движение пакета продолжится с правила, следующего за правилом, вызвавшим переход. Если к пакету во вложенной цепочке будет применено действие ACCEPT, то автоматически пакет будет считаться принятым и в вызывающей цепочке и уже не будет продолжать движение по вызывающим цепочкам. Однако пакет пойдет по другим цепочкам в других таблицах. Дополнительную информацию о порядке прохождения цепочек и таблиц вы сможете получить в главе Порядок прохождения таблиц и цепочек.

Действие - это предопределенная команда, описывающая действие, которое необходимо выполнить, если пакет совпал с заданным критерием. Например, можно применить действие **DROP** или **ACCEPT** к пакету, в зависимости от наших нужд. Существует и ряд других действий, которые описываются ниже в этом разделе. В результате выполнения одних действий, пакет прекращает свое прохождение по цепочке, например **DROP** и **ACCEPT**, в результате других, после выполнения неких операций, продолжает проверку, например, **LOG**, в результате работы третьих даже видоизменяется, например **DNAT** и **SNAT**, **TTL** и **TOS**, но так же продолжает продвижение по цепочке.

6.5.1. Действие ACCEPT

Данная операция не имеет дополнительных ключей. Если над пакетом выполняется действие **ACCEPT**, то пакет прекращает движение по цепочке (и всем вызвавшим цепочкам, если текущая цепочка была вложенной) и считается **ПРИНЯТЫМ** (то бишь пропускается), тем не менее, пакет продолжит движение по цепочкам в других таблицах и может быть отвергнут там. Действие задается с помощью ключа **-j ACCEPT**.

6.5.2. Действие DNAT

DNAT (*Destination Network Address Translation*) используется для преобразования адреса места назначения в IP заголовке пакета. Если пакет подпадает под критерий правила, выполняющего **DNAT**, то этот пакет, и все последующие пакеты из этого же потока, будут подвергнуты преобразованию адреса назначения и переданы на требуемое устройство, хост или сеть. Данное действие может, к примеру, успешно использоваться для предоставления доступа к вашему web-серверу, находящемуся в локальной сети, и не имеющему реального IP адреса. Для этого вы строите правило, которое перехватывает пакеты, идущие на HTTP порт брандмауэра и выполняя **DNAT** передаете их на локальный адрес web-сервера. Для этого действия так же можно указать диапазон адресов, тогда выбор адреса назначения для каждого нового потока будет производиться случайным образом.

Действие **DNAT** может выполняться только в цепочках **PREROUTING** и **OUTPUT** таблицы **nat**, и во вложенных под-цепочках. Важно запомнить, что вложенные подцепочки, реализующие **DNAT** не должны вызываться из других цепочек, кроме **PREROUTING** и **OUTPUT**. **Таблица 6-16. Действие DNAT Ключ --to-destination**

Пример `iptables -t nat -A PREROUTING -p tcp -d 15.45.23.67 --dport 80 -j DNAT --to-destination 192.168.1.1-192.168.1.10`

Описание Ключ **--to-destination** указывает, какой IP адрес должен быть подставлен в качестве адреса места назначения. В выше приведенном примере во всех пакетах, пришедших на адрес 15.45.23.67, адрес назначения будет изменен на один из диапазона от 192.168.1.1 до 192.168.1.10. Как уже указывалось выше, все пакеты из одного потока будут направляться на один и тот же адрес, а для каждого нового потока будет выбираться один из адресов в указанном диапазоне случайным образом. Можно также определить единственный IP адрес. Можно дополнительно указать порт или диапазон портов, на который (которые) будет перенаправлен траффик. Для этого после **ip** адреса через двоеточие укажите порт, например **--to-destination 192.168.1.1:80**, а указание диапазона портов выглядит так: **--to-destination 192.168.1.1:80-100**. Как вы можете видеть, синтаксис действий **DNAT** и **SNAT** во многом схож. Не забывайте, что указание портов допускается только при работе с протоколом **TCP** или **UDP**, при наличии опции **--protocol** в критерии.

Действие **DNAT** достаточно сложно в использовании и требует дополнительного пояснения. Рассмотрим простой пример. У нас есть WEB сервер и мы хотим разрешить доступ к нему из Интернет. Мы имеем только один реальный IP адрес, а WEB-сервер расположен в локальной сети. Реальный IP адрес **\$INET_IP** назначен брандмауэру, HTTP сервер имеет локальный адрес **\$HTTP_IP** и, наконец брандмауэр имеет локальный адрес **\$LAN_IP**. Для начала добавим простое правило в цепочку **PREROUTING** таблицы **nat**:

```
iptables -t nat -A PREROUTING --dst $INET_IP -p tcp --dport 80 -j DNAT --to-destination $HTTP_IP
```

В соответствии с этим правилом, все пакеты, поступающие на 80-й порт адреса \$INET_IP перенаправляются на наш внутренний WEB-сервер. Если теперь обратиться к WEB-серверу из Интернет, то все будет работать прекрасно. Но что же произойдет, если попробовать соединиться с ним из локальной сети? Соединение просто не установится. Давайте посмотрим как маршрутизируются пакеты, идущие из Интернет на наш WEB-сервер. Для простоты изложения примем адрес клиента в Интернет равным \$EXT_BOX.

1. Пакет покидает клиентский узел с адресом \$EXT_BOX и направляется на \$INET_IP
2. Пакет приходит на наш брандмауэр.
3. Брандмауэр, в соответствии с вышеприведенным правилом, подменяет адрес назначения и передает его дальше, в другие цепочки.
4. Пакет передается на \$HTTP_IP.
5. Пакет поступает на HTTP сервер и сервер передает ответ через брандмауэр, если в таблице маршрутизации он обозначен как шлюз для \$EXT_BOX. Как правило, он назначается шлюзом по-умолчанию для HTTP сервера.
6. Брандмауэр производит обратную подстановку адреса в пакете, теперь все выглядит так, как будто бы пакет был сформирован на брандмауэре.
7. Пакет передается клиенту \$EXT_BOX. А теперь посмотрим, что произойдет, если запрос посыпается с узла, расположенного в той же локальной сети. Для простоты изложения примем адрес клиента в локальной сети равным \$LAN_BOX.
8. Пакет покидает \$LAN_BOX.
9. Поступает на брандмауэр.
- 10.Производится подстановка адреса назначения, однако адрес отправителя не подменяется, т.е. исходный адрес остается в пакете без изменения.
- 11.Пакет покидает брандмауэр и отправляется на HTTP сервер.
- 12.HTTP сервер, готовясь к отправке ответа, обнаруживает, что клиент находится в локальной сети (поскольку пакет запроса содержал оригиналный IP адрес, который теперь превратился в адрес назначения) и поэтому отправляет пакет непосредственно на \$LAN_BOX.
- 13.Пакет поступает на \$LAN_BOX. Клиент "путается", поскольку ответ пришел не с того узла, на который отправлялся запрос. Поэтому клиент "сбрасывает" пакет ответа и продолжает ждать "настоящий" ответ.

Проблема решается довольно просто с помощью SNAT. Ниже приводится правило, которое выполняет эту функцию. Это правило вынуждает HTTP сервер передавать ответы на наш брандмауэр, которые затем будут переданы клиенту.

```
iptables -t nat -A POSTROUTING -p tcp --dst $HTTP_IP --dport 80 -j SNAT  
--to-source $LAN_IP
```

Запомните, цепочка POSTROUTING обрабатывается самой последней и к этому моменту пакет уже прошел процедуру преобразования DNAT, поэтому критерий строится на базе адреса назначения \$HTTP_IP.

Если вы думаете, что на этом можно остановиться, то вы ошибаетесь! Представим себе ситуацию, когда в качестве клиента выступает сам брандмауэр. Тогда, к сожалению, пакеты будут передаваться на локальный порт с номером 80 самого брандмауэра, а не на \$HTTP_IP. Чтобы разрешить эту проблему, добавим правило:

```
iptables -t nat -A OUTPUT --dst $INET_IP -p tcp --dport 80 -j DNAT  
--to-destination $HTTP_IP
```

Теперь никаких проблем, с доступом к нашему WEB-серверу, уже не должно возникать.

Каждый должен понять, что эти правила предназначены только лишь для корректной обработки адресации пакетов. В дополнение к этим правилам вам может потребоваться написать дополнительные правила для цепочки FORWARD таблицы filter. Не забудьте при этом, что пакеты уже прошли цепочку PREROUTING и поэтому их адреса назначения уже изменены действием DNAT.

6.5.3. Действие DROP

Данное действие просто "сбрасывает" пакет и `iptables` "забывает" о его существовании. "Сброшенные" пакеты прекращают свое движение полностью, т.е. они не передаются в другие таблицы, как это происходит в случае с действием `ACCEPT`. Следует помнить, что данное действие может иметь негативные последствия, поскольку может оставлять незакрытые "мертвые" сокеты как на стороне сервера, так и на стороне клиента, наилучшим способом защиты будет использование действия `REJECT` особенно при защите от сканирования портов.

6.5.4. Действие LOG

`LOG` -- действие, которое служит для журналирования отдельных пакетов и событий. В журнал могут заноситься заголовки IP пакетов и другая интересующая вас информация. Информация из журнала может быть затем прочитана с помощью `dmesg` или `syslogd` либо с помощью других программ.

Превосходное средство для отладки ваших правил. Неплохо было бы на период отладки правил вместо действия `DROP` использовать действие `LOG`, чтобы до конца убедиться, что ваш брандмауэр работает безупречно. Обратите ваше внимание так же на действие `ULOG`, которое наверняка заинтересует вас своими возможностями, поскольку позволяет выполнять запись журналируемой информации не в системный журнал, а в базу данных MySQL и т.п..

Обратите внимание - если у вас имеются проблемы с записью в системный журнал, то это проблемы не `iptables` или `netfilter`, а `syslogd`. За информацией по конфигурированию `@syslogd` обращайтесь к `man syslog.conf`.

Действие `LOG` имеет пять ключей, которые перечислены ниже. **Таблица 6-17. Ключи действия LOG**

Ключ

Пример `iptables -A FORWARD -p tcp -j LOG --log-level debug`

Описание Используется для задания уровня журналирования (`log level`). Полный список уровней вы найдете в руководстве (`man`) по `syslog.conf`. Обычно, можно задать следующие уровни: `debug`, `info`, `notice`, `warning`, `warn`, `err`, `error`, `crit`, `alert`, `emerg` и `panic`. Ключевое слово `error` означает то же самое, что и `err`, `warn` - `warning` и `panic` - `emerg`. Важно: в последних трех парах слов не следует использовать `error`, `warn` и `panic`. Приоритет определяет различия в том как будут заноситься сообщения в журнал. Все сообщения заносятся в журнал средствами ядра. Если вы установите строку `kern.=info /var/log/iptables` в файле `syslog.conf`, то все ваши сообщения из `iptables`, использующие уровень `info`, будут заноситься в файл `/var/log/iptables`. Однако, в этот файл попадут и другие сообщения, поступающие из других подсистем, которые используют уровень `info`. За дополнительной информацией по `@syslog @` и `@syslog.conf @` рекомендую обращаться к `manpages` и `HOWTO`.

Ключ `--log-prefix`

Пример `iptables -A INPUT -p tcp -j LOG --log-prefix "INPUT packets"`

Описание Ключ задает текст (префикс), которым будут предваряться все сообщения `iptables`. Сообщения со специфичным префиксом затем легко можно найти, к примеру, с помощью `grep`. Префикс может содержать до 29 символов, включая и пробелы.

Ключ `--log-tcp-sequence`

Пример `iptables -A INPUT -p tcp -j LOG --log-tcp-sequence`

Описание Этот ключ позволяет заносить в журнал номер TCP Sequence пакета. Номер TCP Sequence идентифицирует каждый пакет в потоке и определяет порядок "сборки" потока. Этот ключ потенциально опасен для безопасности системы, если системный журнал разрешает доступ "НА ЧТЕНИЕ" всем пользователям. Как и любой другой журнал, содержащий сообщения от `iptables`.

Ключ `--log-tcp-options`

Пример `iptables -A FORWARD -p tcp -j LOG --log-tcp-options`

Описание Этот ключ позволяет заносить в системный журнал различные сведения из заголовка TCP пакета. Такая возможность может быть полезна при отладке. Этот ключ не имеет дополнительных параметров, как и большинство ключей действия LOG.

Ключ --log-ip-options

Пример iptables -A FORWARD -p tcp -j LOG --log-ip-options

Описание Этот ключ позволяет заносить в системный журнал различные сведения из заголовка IP пакета. Во многом схож с ключом --log-tcp-options, но работает только с IP заголовком.

6.5.5. Действие MARK

Используется для установки меток для определенных пакетов. Это действие может выполняться только в пределах таблицы mangle. Установка меток обычно используется для нужд маршрутизации пакетов по различным маршрутам, для ограничения трафика и т.п.. За дополнительной информацией вы можете обратиться к [Linux Advanced Routing and Traffic Control HOW-TO](#). Не забывайте, что "метка" пакета существует только в период времени пока пакет не покинул брандмауэр, т.е. метка не передается по сети. Если необходимо как-то пометить пакеты, чтобы использовать маркировку на другой машине, то можете попробовать манипулировать битами поля TOS. **Таблица 6-18. Ключи действия MARK** Ключ --set-mark

Пример iptables -t mangle -A PREROUTING -p tcp --dport 22 -j MARK --set-mark 2

Описание Ключ --set-mark устанавливает метку на пакет. После ключа --set-mark должно следовать целое беззнаковое число.

6.5.6. Действие MASQUERADE

Маскарадинг (MASQUERADE) в основе своей представляет то же самое, что и SNAT только не имеет ключа --to-source. Причиной тому то, что маскарадинг может работать, например, с dialup подключением или DHCP, т.е. в тех случаях, когда IP адрес присваивается устройству динамически. Если у вас имеется динамическое подключение, то нужно использовать маскарадинг, если же у вас статическое IP подключение, то бесспорно лучшим выходом будет использование действия SNAT.

Маскарадинг подразумевает получение IP адреса от заданного сетевого интерфейса, вместо прямого его указания, как это делается с помощью ключа --to-source в действии SNAT. Действие MASQUERADE имеет хорошее свойство - "забывать" соединения при остановке сетевого интерфейса. В случае же SNAT, в этой ситуации, в таблице трассировщика остаются данные о потерянных соединениях, и эти данные могут сохраняться до суток, поглощая ценную память. Эффект "забывчивости" связан с тем, что при остановке сетевого интерфейса с динамическим IP адресом, есть вероятность на следующем запуске получить другой IP адрес, но в этом случае любые соединения все равно будут потеряны, и было бы глупо хранить трассировочную информацию.

Как вы уже поняли, действие MASQUERADE может быть использовано вместо SNAT, даже если вы имеете постоянный IP адрес, однако, невзирая на положительные черты, маскарадинг не следует считать предпочтительным в этом случае, поскольку он дает большую нагрузку на систему.

Действие MASQUERADE допускается указывать только в цепочке POSTROUTING таблицы nat, так же как и действие SNAT. MASQUERADE имеет ключ, описываемый ниже, использование которого необязательно. **Таблица 6-19. Действие MASQUERADE** Ключ --to-ports

Пример iptables -t nat -A POSTROUTING -p TCP -j MASQUERADE --to-ports 1024-31000

Описание Ключ --to-ports используется для указания порта источника или диапазона портов исходящего пакета. Можно указать один порт, например: --to-ports 1025, или диапазон портов как здесь: --to-ports 1024-3000. Этот ключ можно использовать только в правилах, где критерий содержит явное указание на протокол TCP или UDP с помощью ключа --protocol.

6.5.7. Действие MIRROR

Действие **MIRROR** может использоваться вами только для экспериментов и в демонстрационных целях, поскольку это действие может привести к "зацикливанию" пакета и в результате к "Отказу от обслуживания". В результате действия MIRROR в пакете, поля **source** и **destination** меняются местами (*invert the source and destination fields*) и пакет отправляется в сеть.

Использование этой команды может иметь весьма забавный результат, наверное, со стороны довольно потешно наблюдать, как какой нибудь кульхацкер пытается "взломать" свой собственный компьютер!

Данное действие допускается использовать только в цепочках **INPUT**, **FORWARD** и **PREROUTING**, и в цепочках, вызываемых из этих трех. Пакеты, отправляемые в сеть действием **MIRROR** больше не подвергаются фильтрации, трассировке или NAT, избегая тем самым "зацикливания" и других неприятностей. Однако это не означает, что проблем с этим действием нет. Давайте, к примеру, представим, что на хосте, использующем действие **MIRROR** фабрикуется пакет, с TTL равным 255, на этот же самый хост и пакет подпадает под критерий "зеркалирующего" правила. Пакет "отражается" на этот же хост, а поскольку между "приемником" и "передатчиком" только 1 хоп (hop) то пакет будет прыгать туда и обратно 255 раз. Неплохо для крякера, ведь, при величине пакета 1500 байт, мы потеряем до 380 Кбайт трафика!

6.5.8. Действие QUEUE

Действие **QUEUE** ставит пакет в очередь на обработку пользовательскому процессу. Оно может быть использовано для нужд учета, проксирования или дополнительной фильтрации пакетов.

От переводчика: Далее автор пространно рассуждает о том, что обсуждение данной темы далеко выходит за рамки документа и пр., поэтому, не мудрствуя лукаво, приведу здесь выдержку из http://antonio.mccinet.ru/protection/iptables_howto.html в переводе Евгения Данильченко aka virii5, eugene@kriljon.ru:

"...Для того чтобы эта цель была полезна, необходимы еще два компонента:

1. "queue handler" - обработчик очереди, который выполняет работу по передаче пакетов между ядром и пользовательским приложением; и
2. пользовательское приложение которое будет получать, возможно обрабатывать, и решать судьбу пакетов.

Стандартный обработчик очереди для IPv4 - модуль **ip-queue**, который распространяется с ядром и помечен как экспериментальный. Ниже дан пример, как можно использовать **iptables** для передачи пакетов в пользовательское приложение:

```
# modprobe iptable_filter
# modprobe ip_queue
# iptables -A OUTPUT -p icmp -j QUEUE
```

С этим правилом, созданные локально пакеты ICMP типа (такие, что создаются скажем при помощи команды **ping**) попадают в модуль **ip-queue**, который затем пытается передать их в пользовательское приложение. Если ни одно из таких приложений не найдено, пакеты сбрасываются. Чтобы написать пользовательскую программу обработки пакетов, используйте **libipq API**. Оно распространяется с пакетом **iptables**. Примеры можно найти в **testsuite tools** (например **redirect.c**) на CVS. Статус **ip-queue** можно проверить с помощью: **/proc/net/ip_queue** Максимальную длинну очереди (то есть, число пакетов передаваемых в пользовательское приложение без подтверждения обработки) можно контролировать с помощью: **/proc/sys/net/ipv4/ip_queue_maxlen** По умолчанию - максимальная длина очереди равна 1024. Как только этот предел достигается, новые пакеты будут сбрасываться, пока очередь не снизиться ниже данного предела. Хорошие протоколы, такие как TCP интерпретируют сброшенные пакеты как перегруженность канала передачи, и успешно с этим справляются (насколько я помню, пакет будет просто переслан заново удаленной стороной, прим. перевод.). Однако, может потребоваться некоторого рода

экспериментирование, чтобы определить оптимальную длину очереди в каждом конкретном случае, если по умолчанию очередь слишком мала..."

6.5.9. Действие REDIRECT

Выполняет перенаправление пакетов и потоков на другой порт той же самой машины. К примеру, можно пакеты, поступающие на HTTP порт перенаправить на порт HTTP proxy. Действие REDIRECT очень удобно для выполнения "прозрачного" проксирования (*transparent proxying*), когда машины в локальной сети даже не подозревают о существовании прокси.

REDIRECT может использоваться только в цепочках PREROUTING и OUTPUT таблицы nat. И конечно же это действие можно выполнять в подцепочках, вызываемых и вышеуказанных. Для действия REDIRECT предусмотрен только один ключ.

Таблица 6-20. Действие REDIRECT Ключ --to-ports

Пример iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to-ports 8080

Описание Ключ **--to-ports** определяет порт или диапазон портов назначения. Без указания ключа **--to-ports**, перенаправления не происходит, т.е. пакет идет на тот порт, куда и был назначен. В примере, приведенном выше, **--to-ports 8080** указан один порт назначения. Если нужно указать диапазон портов, то мы должны написать нечто подобное **--to-ports 8080-8090**. Этот ключ можно использовать только в правилах, где критерий содержит явное указание на протокол TCP или UDP с помощью ключа **--protocol**.

6.5.10. Действие REJECT

REJECT используется, как правило, в тех же самых ситуациях, что и DROP, но в отличие от DROP, команда REJECT выдает сообщение об ошибке на хост, передавший пакет. Действие REJECT на сегодняшний день может использоваться только в цепочках INPUT, FORWARD и OUTPUT (и во вложенных в них цепочках). Пока существует только единственный ключ, управляющий поведением команды REJECT.

Таблица 6-21. Действие REJECT Ключ --reject-with

Пример iptables -A FORWARD -p TCP --dport 22 -j REJECT --reject-with tcp-reset

Описание Указывает, какое сообщение необходимо передать в ответ, если пакет совпал с заданным критерием. При применении действия REJECT к пакету, сначала на хост-отправитель будет отослан указанный ответ, а затем пакет будет "брошен". Допускается использовать следующие типы ответов: **icmp-net-unreachable**, **icmp-host-unreachable**, **icmp-port-unreachable**, **icmp-proto-unreachable**, **icmp-net-prohibited** и **icmp-host-prohibited**. По-умолчанию передается сообщение **port-unreachable**. Все вышеуказанные типы ответов являются ICMP error messages. Дополнительную информацию по типам ICMP сообщений вы можете получить в приложении *Типы ICMP*. В заключение укажем еще один тип ответа - **tcp-reset**, который используется только для протокола TCP. Если указано значение **tcp-reset**, то действие REJECT передаст в ответ пакет TCP RST, пакеты TCP RST используются для закрытия TCP соединений. За дополнительной информацией обращайтесь к *RFC 793 - Transmission Control Protocol*. (Список типов ICMP ответов и их алиасов вы сможете получить введя команду **iptables -j REJECT -h** прим. перев.).

6.5.11. Действие RETURN

Действие RETURN прекращает движение пакета по текущей цепочке правил и производит возврат в вызывающую цепочку, если текущая цепочка была вложенной, или, если текущая цепочка лежит на самом верхнем уровне (например INPUT), то к пакету будет применена политика по-умолчанию. Обычно, в качестве политики по-умолчанию назначают действия ACCEPT или DROP .

Для примера, допустим, что пакет идет по цепочке INPUT и встречает правило, которое производит переход во вложенную цепочку `- - -j` `mp EXAMPLE_CHAIN`. Далее, в цепочке EXAMPLE_CHAIN пакет встречает правило, которое выполняет действие `- - j` `mp RETURN`. Тогда произойдет возврат пакета в цепочку INPUT. Другой пример, пусть пакет встречает правило, которое выполняет действие `- - j` `mp RETURN` в цепочке INPUT. Тогда к пакету будет применена политика по-умолчанию цепочки INPUT.

6.5.12. Действие SNAT

SNAT используется для преобразования сетевых адресов (Source Network Address Translation), т.е. изменение исходящего IP адреса в IP заголовке пакета. Например, это действие можно использовать для предоставления выхода в Интернет другим компьютерам из локальной сети, имея лишь один уникальный IP адрес. Для этого необходимо включить пересылку пакетов (forwarding) в ядре и затем создать правила, которые будут транслировать исходящие IP адреса нашей локальной сети в реальный внешний адрес. В результате, внешний мир ничего не будет знать о нашей локальной сети, он будет считать, что запросы пришли с нашего брандмауэра.

SNAT допускается выполнять только в таблице nat, в цепочке POSTROUTING. Другими словами, только здесь допускается преобразование исходящих адресов. Если первый пакет в соединении подвергся преобразованию исходящего адреса, то все последующие пакеты, из этого же соединения, будут преобразованы автоматически и не пойдут через эту цепочку правил. **Таблица 6-22. Действие SNAT**
Ключ `- - to-source`

Пример `iptables -t nat -A POSTROUTING -p tcp -o eth0 -j SNAT --to-source 194.236.50.155-194.236.50.160:1024-32000`

Описание Ключ `- - to-source` используется для указания адреса, присваиваемого пакету. Все просто, вы указываете IP адрес, который будет подставлен в заголовок пакета в качестве исходящего. Если вы собираетесь перераспределять нагрузку между несколькими брандмауэрами, то можно указать диапазон адресов, где начальный и конечный адреса диапазона разделяются дефисом, например: 194.236.50.155-194.236.50.160. Тогда, конкретный IP адрес будет выбираться из диапазона случайным образом для каждого нового потока. Дополнительно можно указать диапазон портов, которые будут использоваться только для нужд SNAT. Все исходящие порты будут после этого перекартироваться в заданный диапазон. iptables старается, по-возможности, избегать перекартирования портов, однако не всегда это возможно, и тогда производится перекартирование. Если диапазон портов не задан, то исходные порты ниже 512 перекартируются в диапазоне 0-511, порты в диапазоне 512-1023 перекартируются в диапазоне 512-1023, и, наконец порты из диапазона 1024-65535 перекартируются в диапазоне 1024-65535. Что касается портов назначения, то они не подвергаются перекартированию.

6.5.13. Действие TOS

Команда TOS используется для установки битов в поле Type of Service IP заголовка. Поле TOS содержит 8 бит, которые используются для маршрутизации пакетов. Это один из нескольких полей, используемых iproute2. Так же важно помнить, что данное поле может обрабатываться различными маршрутизаторами с целью выбора маршрута движения пакета. Как уже указывалось выше, это поле, в отличие от MARK, сохраняет свое значение при движении по сети, а поэтому может использоваться вами для маршрутизации пакета. На сегодняшний день, большинство маршрутизаторов в Интернете никак не обрабатывают это поле, однако есть и такие, которые смотрят на него. Если вы используете это поле в своих нуждах, то подобные маршрутизаторы могут принять неверное решение при выборе маршрута, поэтому, лучше всего использовать это поле для своих нужд только в пределах вашей WAN или LAN.

Внимание!

Действие TOS воспринимает только предопределенные числовые значения и мнемоники, которые вы можете найти в `linux/ip.h`. Если вам действительно необходимо устанавливать произвольные значения в поле TOS, то можно воспользоваться "заплатой" FTOS с сайта [Paksecured Linux Kernel patches](#), поддерживаемого Matthew G. Marsh. Однако, будьте крайне осторожны с этой "заплатой". Не следует использовать нестандартные значения TOS иначе как в особых ситуациях.

Данное действие допускается выполнять только в пределах таблицы `mangle`.

В некоторых старых версиях `iptables` (1.2.2 и ниже) это действие реализовано с ошибкой (не исправляется контрольная сумма пакета), а это ведет к нарушению протокола обмена и в результате такие соединения обрываются.

Команда `TOS` имеет только один ключ, который описан ниже. **Таблица 6-23. Действие TOS Ключ `--set-tos`**

Пример `iptables -t mangle -A PREROUTING -p TCP --dport 22 -j TOS --set-tos 0x10`

Описание Ключ `--set-tos` определяет числовое значение в десятичном или шестнадцатиричном виде. Поскольку поле `TOS` является 8-битным, то вы можете указать число в диапазоне от 0 до 255 (0x00 - 0xFF). Однако, большинство значений этого поля никак не используются. Вполне возможно, что в будущих реализациях TCP/IP числовые значения могут быть изменены, поэтому, во избежание ошибок, лучше использовать мнемонические обозначения: `Minimize-Delay` (16 или 0x10), `Maximize-Throughput` (8 или 0x08), `Maximize-Reliability` (4 или 0x04), `Minimize-Cost` (2 или 0x02) или `Normal-Service` (0 или 0x00). По умолчанию большинство пакетов имеют признак `Normal-Service`, или 0. Список мнемоник вы сможете получить, выполнив команду `iptables -h TOS`.

6.5.14. Действие TTL

Действие `TTL` используется для изменения содержимого поля `Time To Live` в IP заголовке. Один из вариантов применения этого действия - это устанавливать значение поля `Time To Live` во всех исходящих пакетах в одно и то же значение. Для чего это?! Есть некоторые провайдеры, которые очень не любят, когда одним подключением пользуется несколько компьютеров, если мы начинаем устанавливать на все пакеты одно и то же значение `TTL`, то тем самым мы лишаем провайдера одного из критериев определения того, что подключение к Интернету разделяется несколькими компьютерами. Для примера можно привести число `TTL = 64`, которое является стандартным для ядра Linux.

За дополнительной информацией по установке значения по умолчанию обращайтесь к [ip-sysctl.txt](#), который вы найдете в приложении [Ссылки на другие ресурсы](#).

Действие `TTL` можно указывать только в таблице `mangle` и нигде больше. Для данного действия предусмотрено 3 ключа, описываемых ниже.

Таблица 6-24. Действие TTL |Ключ | `--ttl-set`| |Пример |`iptables -t mangle -A PREROUTING -i eth0 -j TTL --ttl-set 64`| |Описание | Устанавливает поле `TTL` в заданное значение. Оптимальным считается значение около 64. Это не слишком много, но и не слишком мало. Не задавайте слишком большое значение, это может иметь неприятные последствия для вашей сети. Представьте себе, что пакет "зацикливается" между двумя неправильно сконфигурированными роутерами, тогда, при больших значениях `TTL`, есть риск "потерять" значительную долю пропускной способности канала.| |Ключ | `--ttl-dec`| |Пример |`iptables -t mangle -A PREROUTING -i eth0 -j TTL --ttl-dec 1`| |Описание |

Уменьшает значение поля `TTL` на заданное число. Например, пусть входящий пакет имеет значение `TTL` равное 53 и мы выполняем команду `--ttl-dec 3`, тогда пакет покинет наш хост с полем `TTL` равным 49. Не забывайте, что сетевой код автоматически уменьшит значение `TTL` на 1, поэтому, фактически мы получаем $53 - 3 - 1 = 49$.| |Ключ | `--ttl-inc`| |Пример |`iptables -t mangle -A PREROUTING -i eth0 -j TTL --ttl-inc 1`| |Описание | Увеличивает значение поля `TTL` на заданное число. Возьмем предыдущий пример, пусть к нам поступает пакет с `TTL = 53`, тогда, после выполнения команды `--ttl-inc 4`, на выходе с нашего хоста, пакет будет иметь `TTL = 56`, не забывайте об автоматическом уменьшении поля `TTL` сетевым кодом ядра, т.е. фактически мы получаем выражение $53 + 4 - 1 = 56$. Увеличение поля `TTL` может использоваться для того, чтобы сделать наш брандмауэр менее "заметным" для трассировщиков (traceroutes). Программы трассировки любят за ценную информацию при поиске проблемных участков сети, и ненавидят за это же, поскольку эта информация может использоваться крашерами в неблаговидных целях. Пример использования вы можете найти в сценарии [ttl-inc.txt](#).

6.5.15. Действие ULOG

Действие **ULOG** предоставляет возможность журналирования пакетов в пользовательское пространство. Оно заменяет традиционное действие **LOG**, базирующееся на системном журнале. При использовании этого действия, пакет, через сокеты netlink, передается специальному демону который может выполнять очень детальное журналирование в различных форматах (обычный текстовый файл, база данных MySQL и пр.) и к тому же поддерживает возможность добавления надстроек (плагинов) для формирования различных выходных форматов и обработки сетевых протоколов. Пользовательскую часть ULOGD вы можете получить на домашней странице [ULOGD project page](#). **Таблица 6-25.**

Действие ULOG Ключ --ulog-nlgroup

Пример iptables -A INPUT -p TCP --dport 22 -j ULOG --ulog-nlgroup 2

Описание Ключ **--ulog-nlgroup** сообщает ULOG в какую группу netlink должен быть передан пакет. Всего существует 32 группы (от 1 до 32). Если вы желаете передать пакет в 5-ю группу, то можно просто указать **--ulog-nlgroup 5**. По-умолчанию используется 1-я группа.

Ключ --ulog-prefix

Пример iptables -A INPUT -p TCP --dport 22 -j ULOG --ulog-prefix "SSH connection attempt: "

Описание Ключ **--ulog-prefix** имеет тот же смысл, что и аналогичная опция в действии **LOG**. Длина строки префикса не должна превышать 32 символа

Ключ --ulog-cprange

Пример iptables -A INPUT -p TCP --dport 22 -j ULOG --ulog-cprange 100

Описание Ключ **--ulog-cprange** определяет, какую долю пакета, в байтах, надо передавать демону **ULOG**. Если указать число 100, как показано в примере, то демону будет передано только 100 байт из пакета, это означает, что демону будет передан заголовок пакета и некоторая часть области данных пакета. Если указать 0, то будет передан весь пакет, независимо от его размера. Значение по-умолчанию равно 0.

Ключ --ulog-qthreshold

Пример iptables -A INPUT -p TCP --dport 22 -j ULOG --ulog-qthreshold 10

Описание Ключ **--ulog-qthreshold** устанавливает величину буфера в области ядра. Например, если задать величину буфера равной 10, как в примере, то ядро будет накапливать журналируемые пакеты во внутреннем буфере и передавать в пользовательское пространство группами по 10 пакетов. По-умолчанию размер буфера равен 1 из-за сохранения обратной совместимости с ранними версиями ulogd, которые не могли принимать группы пакетов.

Глава 7. Файл rc.firewall

В этой главе мы рассмотрим настройку брандмауэра на примере сценария [rc.firewall.txt](#). Мы будем брать каждую базовую настройку и рассматривать как она работает и что делает. Это может натолкнуть вас на решение ваших собственных задач. Для запуска этого сценария вам потребуется внести в него изменения таким образом, чтобы он мог работать с вашей конфигурацией сети, в большинстве случаев достаточно изменить только переменные.

Note

Примечательно, что есть более эффективные способы задания наборов правил, однако я исходил из мысли о большей удобочитаемости сценария, так, чтобы каждый смог понять его без глубоких познаний оболочки BASH.

7.1. Пример rc.firewall

Итак, все готово для разбора файла примера [rc.firewall.txt](#) (сценарий включен в состав данного документа в приложении *Примеры сценариев*). Он достаточно велик, но только из-за большого количества комментариев. Сейчас я предлагаю вам просмотреть этот файл, чтобы получить представление о его содержимом и затем вернуться сюда за более подробными пояснениями.

7.2. Описание сценария rc.firewall

7.2.1. Конфигурация

Первая часть файла [rc.firewall.txt](#) является конфигурационным разделом. Здесь задаются основные настройки брандмауэра, которые зависят от вашей конфигурации сети. Например IP адреса - наверняка должны быть изменены на ваши собственные. Переменная \$INET_IP должна содержать реальный IP адрес, если вы подключаетесь к Интернет через DHCP, то вам следует обратить внимание на скрипт [rc.DHCP.firewall.txt](#). Аналогично \$INET_IFACE должна указывать ваше устройство, через которое осуществляется подключение к Интернет. Это может быть, к примеру, eth0, eth1, ppp0, tr0 и пр.

Этот сценарий не содержит каких либо настроек, специфичных для DHCP, PPPoE, поэтому эти разделы не заполнены. То же самое касается и других "пустых" разделов. Это сделано преднамеренно, чтобы вы могли более наглядно видеть разницу между сценариями. Если вам потребуется заполнить эти разделы, то вы можете взять их из других скриптов, или написать свой собственный.

Раздел *Local Area Network* должен содержать настройки, соответствующие конфигурации вашей локальной сети. Вы должны указать локальный IP адрес брандмауэра, интерфейс, подключенный к локальной сети, маску подсети и широковещательный адрес.

Далее следует секция *Localhost Configuration*, которую изменять вам едва ли придется. В этой секции указывается локальный интерфейс lo и локальный IP адрес 127.0.0.1. За разделом *Localhost Configuration*, следует секция *Iptables Configuration*. Здесь создается переменная \$IPTABLES, содержащая путь к файлу iptables (обычно /usr/local/sbin/iptables). Если вы устанавливали iptables из исходных модулей, то у вас путь к iptables может несколько отличаться от приведенного в сценарии (например /usr/sbin/iptables), однако в большинстве дистрибутивов iptables расположена именно здесь.

7.2.2. Загрузка дополнительных модулей

В первую очередь, командой /sbin/depmod -a, выполняется проверка зависимостей модулей после чего производится подгрузка модулей, необходимых для работы сценария. Страйтесь в ваших сценариях загружать только необходимые модули. Например, по каким то причинам мы собрали поддержку действий LOG, REJECT и MASQUERADE в виде подгружаемых модулей и теперь собираемся строить правила, использующие эти действия, тогда соответствующие модули необходимо загрузить командами:

```
/sbin/insmod ipt_LOG  
/sbin/insmod ipt_REJECT  
/sbin/insmod ipt_MASQUERADE
```

Внимание!

В своих сценариях я принудительно загружаю все необходимые модули, во избежание отказов. Если происходит ошибка во время загрузки модуля, то причин может быть множество, но основной причиной является то, что подгружаемые модули скомпилированы с ядром статически. За дополнительной информацией обращайтесь к разделу *Проблемы загрузки модулей*.

В следующей секции приводится ряд модулей, которые не используются в данном сценарии, но перечислены для примера. Так например модуль `ipt_owner`, который может использоваться для предоставления доступа к сети с вашей машины только определенному кругу пользователей, повышая, тем самым уровень безопасности. Информацию по критериям `ipt_owner`, смотрите в разделе *Критерий Owner* главы *Как строить правила*.

Мы можем загрузить дополнительные модули для проверки "состояния" пакетов (`state matching`). Все модули, расширяющие возможности проверки состояния пакетов, именуются как `ip_conntrack_*` и `ip_nat_*`. С помощью этих модулей осуществляется трассировка соединений по специфичным протоколам. Например: протокол FTP является комплексным протоколом по определению, он передает информацию о соединении в области данных пакета. Так, если наш локальный хост передает через брандмауэр, производящий трансляцию адресов, запрос на соединение с

FTP сервером в Интернет, то внутри пакета передается локальный IP адрес хоста. А поскольку, IP адреса, зарезервированные для локальных сетей, считаются ошибочными в Интернет, то сервер не будет знать что делать с этим запросом, в результате соединение не будет установлено. Вспомогательный модуль FTP NAT выполняет все необходимые действия по преобразованию адресов, поэтому FTP сервер фактически получит запрос на соединение от имени нашего внешнего IP адреса и сможет установить соединение. То же самое происходит при использовании DCC для передачи файлов и чатов. Установка соединений этого типа требует передачи IP адреса и порта по протоколу IRC, который так же проходит через трансляцию сетевых адресов на брандмауэр. Без специального модуля расширения работоспособность протоколов FTP и IRC становится весьма сомнительной. Например, вы можете принимать файлы через DCC, но не можете отправлять. Это обуславливается тем, как DCC "запускает" соединение. Вы сообщаете принимающему узлу о своем желании передать файл и куда он должен подключиться. Без вспомогательного модуля DCC соединение выглядит так, как если бы мы потребовали установление соединения внешнего приемника с узлом в нашей локальной сети, проще говоря такое соединение будет "обрущено". При использовании же вспомогательного модуля все работает прекрасно. поскольку приемнику передается корректный IP адрес для установления соединения.

Если у вас наблюдаются проблемы с прохождением mIRC DCC через брандмауэр, но при этом другие IRC-клиенты работают вполне корректно - прочтайте раздел *Проблемы mIRC DCC* в приложении *Общие проблемы и вопросы*.

Дополнительную информацию по модулям `conntrack` и `nat` читайте в приложении *Общие проблемы и вопросы*. Так же не забывайте о документации, включаемой в пакет `iptables`. Чтобы иметь эти дополнительные возможности, вам потребуется установить `patch-o-matic` и пересобрать ядро. Как это сделать - объясняется выше в главе *Подготовка*.

Заметьте, что загружать модули `ip_nat_irc` и `ip_nat_ftp` вам потребуется только в том случае, если вы хотите, чтобы преобразование сетевых адресов (*Network Adress Translation*) производилось корректно с протоколами FTP и IRC. Так же вам потребуется подгрузить модули `ip_conntrack_irc` и `ip_conntrack_ftp` до загрузки модулей NAT.

7.2.3. Настройка /proc

Здесь мы запускаем пересылку пакетов (*IP forwarding*), записав единицу в файл `/proc/sys/net/ipv4/ip_forward` таким способом:

```
echo "1" > /proc/sys/net/ipv4/ip_forward
```

Внимание!

bq. Наверное стоит задуматься над тем где и когда включать пересылку (*IP forwarding*). В этом и в других сценариях в данном руководстве, мы включаем пересылку до того как создадим какие либо правила `iptables`. От начала работы пересылки (*IP forwarding*) до момента, когда будут созданы необходимые правила, при нашем варианте, может пройти от нескольких миллисекунд до минут, все зависит от объема работы, выполняемой сценарием и быстродействия конкретного компьютера. Понятно, что это дает некоторый промежуток времени, когда злоумышленник может проникнуть через брандмауэр. Поэтому, в реальной ситуации запускать пересылку (*IP forwarding*) следует после создания всего набора правил. Здесь же я поместил включение пересылки в начале исключительно в целях удобочитаемости.

Если вам необходима поддержка динамического IP, (при использовании SLIP, PPP или DHCP) вы можете раскомментарить строку:

```
echo "1" > /proc/sys/net/ipv4/ip_dynaddr
```

Если вам требуется включить любые другие опции, вы должны обращаться к соответствующей документации по этим опциям. Хороший и лаконичный документ по файловой системе `/proc` поставляется вместе с ядром. Ссылки на другие документы вы найдете в приложении *Ссылки на другие ресурсы*.

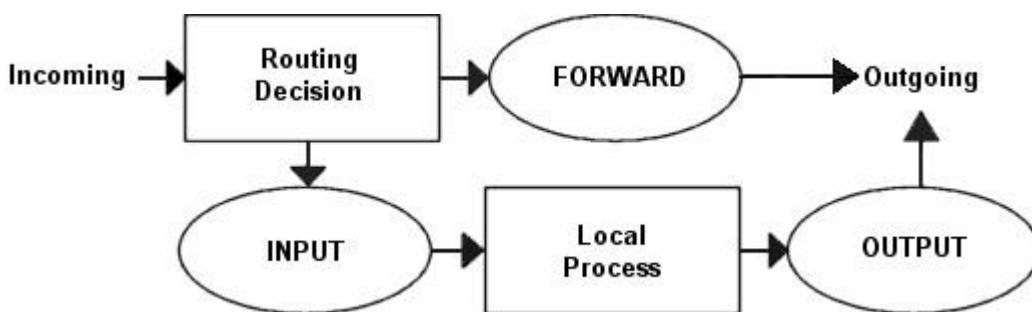
Сценарий [rc.firewall.txt](#) и все остальные сценарии в данном руководстве, содержат

небольшую по размерам секцию не требуемых (*non-required*) настроек /proc. Как бы привлекательно не выглядели эти опции - не включайте их, пока не убедитесь, что достаточно четко представляете себе функции, которые они выполняют.

7.2.4. Размещение правил по разным цепочкам

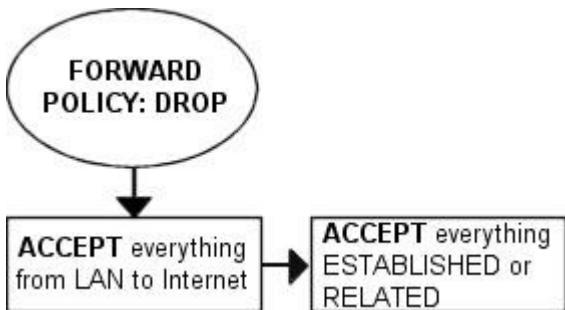
Здесь мы поговорим о пользовательских цепочках, в частности - о пользовательских цепочках, определяемых в сценарии [rc.firewall.txt](#). Мой вариант разделения правил по дополнительным цепочкам может оказаться неприемлемым в том или ином конкретном случае. Я надеюсь, что смогу показать вам возможные "подводные камни". Данный раздел тесно перекликается с главой *Порядок прохождения таблиц и цепочек* и совершенно нeliшним будет еще раз, хотя бы бегло, просмотреть ее.

Распределив набор правил по пользовательским цепочкам, я добился экономии процессорного времени, без потери уровня безопасности системы и читабельности сценариев. Вместо того, чтобы пропускать TCP пакеты через весь набор правил (и для ICMP, и для UDP), я просто отбираю TCP пакеты и пропускаю их через пользовательскую цепочку, предназначенную именно для TCP пакетов, что приводит к уменьшению нагрузки на систему. На следующей картинке схематично приводится порядок прохождения пакетов через *netfilter*. В действительности, эта картинка выглядит несколько ограниченно по сравнению со схемой, приведенной в главе *Порядок прохождения таблиц и цепочек*.



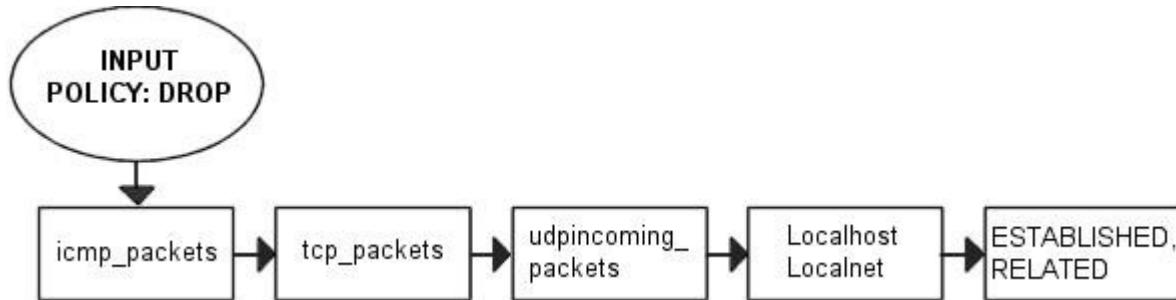
Основное назначение рисунка - освежить нашу память. В целом, данный пример сценария основан на предположении, что мы имеем одну локальную сеть, один брандмауэр (firewall) и единственное подключение к Интернет, с постоянным IP адресом (в противоположность PPP, SLIP, DHCP и прочим). Так же предполагается, что доступ к сервисам Интернет идет через брандмауэр, что мы полностью доверяем нашей локальной сети и поэтому не собираемся блокировать трафик, исходящий из локальной сети, однако Интернет не может считаться доверительной сетью и поэтому необходимо ограничить возможность доступа в нашу локальную сеть извне. Мы собираемся исходить из принципа "Все что не разрешено - то запрещено". Для выполнения последнего ограничения, мы устанавливаем политику по-умолчанию - **DROP**. Тем самым мы отсекаем соединения, которые явно не разрешены.

А теперь давайте рассмотрим что нам нужно сделать и как.



Для начала - позволим соединения из локальной сети с Интернет. Для этого нам потребуется выполнить преобразование сетевых адресов (NAT). Делается это в цепочке **PREROUTING** (Я полагаю, что здесь автор просто допустил опечатку, поскольку в тексте сценария заполняется цепочка **POSTROUTING**, да и мы уже знаем, что SNAT производится в цепочке **POSTROUTING** таблицы **nat** **прим. перев.**), которая заполняется последней в нашем сценарии. Подразумевается, также, выполнение некоторой фильтрации

в цепочке FORWARD. Если мы полностью доверяем нашей локальной сети, пропуская весь траффик в Интернет, то это еще не означает доверия к Интернет и, следовательно необходимо вводить ограничения на доступ к нашим компьютерам извне. В нашем случае мы допускаем прохождение пакетов в нашу сеть только в случае уже установленного соединения, либо в случае открытия нового соединения, но в рамках уже существующего (ESTABLISHED и RELATED).

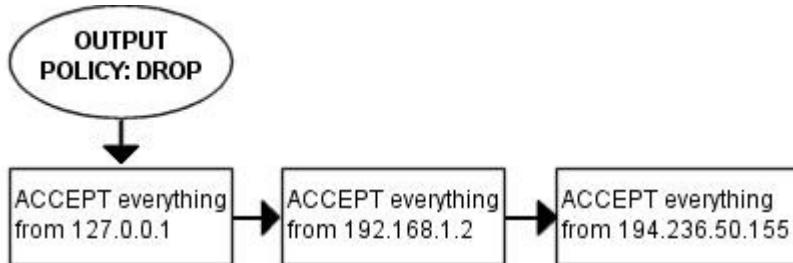


Что касается машины-брандмауэра - необходимо до минимума свести сервисы, работающие с Интернет. Следовательно мы допускаем только HTTP, FTP, SSH и IDENTD доступ к брандмауэру. Все эти протоколы мы будем считать допустимыми в цепочке INPUT, соответственно нам необходимо разрешить "ответный" траффик в цепочке OUTPUT. Поскольку мы предполагаем доверительные взаимоотношения с локальной сетью, то мы добавляем правила для диапазона адресов локальной сети, а заодно и для локального сетевого интерфейса и локального IP адреса (127.0.0.1). Как уже упоминалось выше, существует ряд диапазонов адресов, выделенных специально для локальных сетей, эти адреса считаются в Интернет ошибочными и как правило не обслуживаются. Поэтому и мы запретим любой траффик из Интернет с исходящим адресом, принадлежащим диапазонам локальных сетей. И в заключение прочитайте главу *Общие проблемы и вопросы*.

Так как у нас работает FTP сервер, то правила, обслуживающие соединения с этим сервером, желательно было бы поместить в начало цепочки INPUT, добиваясь тем самым уменьшения нагрузки на систему. В целом же, надо понимать, что чем меньше правил проходит пакет, тем больше экономия процессорного времени, тем ниже нагрузка на систему. С этой целью я разбил набор правил на дополнительные цепочки.

В нашем примере я разбил пакеты на группы по их принадлежности к тому или иному протоколу. Для каждого типа протокола создана своя цепочка правил, например, `tcp_packets`, которая содержит правила для проверки всех допустимых TCP портов и протоколов. Для проведения дополнительной проверки пакетов, прошедших через одну цепочку, может быть создана другая. В нашем случае таковой является цепочка `allowed`. В этой цепочке производится дополнительная проверка отдельных характеристик TCP пакетов перед тем как принять окончательное решение о пропуске. ICMP пакеты следуют через цепочку `icmp_packets`. Здесь мы просто пропускаем все ICMP пакеты с указанным кодом сообщения. И наконец UDP пакеты. Они проходят через цепочку `udr_packets`, которая обрабатывает входящие UDP пакеты. Если они принадлежат допустимым сервисам, то они пропускаются без проведения дополнительной проверки.

Поскольку мы рассматриваем сравнительно небольшую сеть, то наш брандмаэр используется еще и в качестве рабочей станции, поэтому мы делаем возможным соединение с Интернет и с самого брандмауэра.



И в завершение о цепочке OUTPUT. Мы не выполняем каких либо специфичных блокировок для пользователей, однако мы не хотим, чтобы кто либо, используя наш брандмауэр выдавал в сеть "поддельные" пакеты, поэтому мы устанавливаем правила, позволяющие прохождение пакетов только с нашим адресом в локальной сети, с нашим локальным адресом (127.0.0.1) и с нашим адресом в Интернет. С этих адресов пакеты пропускаются цепочкой OUTPUT, все остальные (скорее всего сфальсифицированные) отсекаются политикой по-умолчанию DROP.

7.2.5. Установка политик по-умолчанию

Прежде, чем приступить к созданию набора правил, необходимо определиться с политиками цепочек по-умолчанию. Политика по-умолчанию устанавливается командой, подобной приводимой ниже

```
iptables [-P {chain} {policy}]
```

Политика по-умолчанию представляет собой действие, которое применяется к пакету, не попавшему под действие ни одного из правил в цепочке. (Небольшое уточнение, команда iptables -P применима ТОЛЬКО К ВСТРОЕННЫМ цепочкам, т.е. INPUT, FORWARD, OUTPUT и т.п., и не применима к пользовательским цепочкам. **прим. перев.**).

Будьте предельно осторожны с установкой политик по-умолчанию для цепочек из таблиц, не предназначенных для фильтрации, так как это может приводить к довольно странным результатам.

7.2.6. Создание пользовательских цепочек в таблице filter

Итак, у вас перед глазами наверняка уже стоит картинка движения пакетов через различные цепочки, и как эти цепочки взаимодействуют между собой! Вы уже должны ясно представлять себе цели и назначение данного сценария. Давайте начнем создавать цепочки и наборы правил для них.

Прежде всего необходимо создать дополнительные цепочки с помощью команды -N. Сразу после создания цепочки еще не имеют ни одного правила. В нашем примере создаются цепочки icmp_packets, tcp_packets, udp_packets и цепочка allowed, которая вызывается из цепочки tcp_packets. Входящие пакеты с интерфейса \$INET_IFACE (т.е. из Интернет), по протоколу ICMP перенаправляются в цепочку icmp_packets, пакеты протокола TCP перенаправляются в цепочку tcp_packets и входящие пакеты UDP с интерфейса eth0 идут в цепочку udp_packets. Более подробное описание вы найдете в разделе *Цепочка INPUT*. Синтаксис команды для создания своей цепочки очень прост:

```
iptables [-N chain]
```

7.2.6.1. Цепочка bad_tcp_packets

Эта цепочка предназначена для отфильтровывания пакетов с "неправильными" заголовками и решения ряда других проблем. Здесь отфильтровываются все пакеты, которые распознаются как NEW, но не являются SYN пакетами, а так же обрабатываются SYN/ACK-пакеты, имеющие статус NEW. Эта цепочка может быть использована для защиты от вторжения и сканирования портов. Сюда, так же, добавлено правило для отсеивания пакетов со статусом INVALID.

Если вы пожелаете почитать более подробно об этой проблеме, то смотрите раздел *Пакеты со статусом NEW и со сброшенным битом SYN* в приложении *Общие проблемы и вопросы*. Разумеется, не всегда справедливо будет просто сбрасывать пакеты с признаком NEW и сброшенным битом SYN, но в 99% случаев это оправданный шаг. Поэтому мой сценарий заносит информацию о таких пакетах в читемный журнал, а затем "сбрасывает" их.

Причина, по которой для SYN/ACK-пакетов со статусом NEW @применяется действие @REJECT, достаточно проста. Она описывается в разделе *SYN/ACK - пакеты и пакеты со статусом NEW* приложения *Общие проблемы и вопросы*. Общепринятой считается необходимость отправления пакета RST в подобных случаях (RST в ответ на незапрошенный SYN/ACK). Тем самым мы предотвращаем возможность атаки "Предсказание номера TCP-последовательности" (Sequence Number Prediction) на другие узлы сети.

7.2.6.2. Цепочка allowed

TCP пакет, следуя с интерфейса `$INET_IFACE`, попадает в цепочку `tcp_packets`, если пакет следует на разрешенный порт, то после этого проводится дополнительная проверка в цепочке `allowed`.

Первое правило проверяет, является ли пакет SYN пакетом, т.е. запросом на соединение. Такой пакет мы считаем допустимым и пропускаем. Следующее правило пропускает все пакеты с признаком `ESTABLISHED` или `RELATED`. Когда соединение устанавливается SYN пакетом, и на этот запрос был отправлен положительный ответ, то оно получает статус `ESTABLISHED`. Последним правилом в этой цепочке сбрасываются все остальные TCP пакеты. Под это правило попадают пакеты из несуществующего соединения, пакеты со сброшенным битом SYN, которые пытаются запустить соединение. Не SYN пакеты практически не используются для запуска соединения, за исключением случаев сканирования портов. Насколько я знаю, на сегодняшний день нет реализации TCP/IP, которая поддерживала бы открытие соединения иначе, чем передача SYN пакета, поэтому на 99% можно быть уверенным, что сброшены пакеты, посланные сканером портов.

7.2.6.3. Цепочка для TCP

Итак, мы подошли к TCP соединениям. Здесь мы указываем, какие порты могут быть доступны из Internet. Несмотря на то, что даже если пакет прошел проверку здесь, мы все равно все пакеты передаем в цепочку `allowed` для дополнительной проверки.

Я открыл TCP порт с номером 21, который является портом управления FTP соединениями. и далее, я разрешаю все RELATED соединения, разрешая, тем самым, PASSIVE FTP, при условии, что был загружен модуль `ip_conntrack_ftp`. Если вам потребуется запретить FTP соединения, то вам потребуется выгрузить модуль `ip_conntrack_ftp` и удалить строку `$IPTABLES -A tcp_packets -p TCP -s 0/0 --dport 21 -j allowed` из сценария [rc.firewall.txt](#).

Порт 22 - это SSH, который намного более безопасен чем telnet на 23 порту. Если Вам вздумается предоставить доступ к командной оболочке (shell) кому бы то ни было из Интернет, то лучше конечно пользоваться SSH. Однако, хочу заметить, что вообще-то считается дурным тоном предоставлять доступ к брандмауэру любому кроме вас самих. Ваш сетевой экран должен иметь только те сервисы, которые действительно необходимы и не более того.

Порт 80 - это порт HTTP, другим словами - web сервер, уберите это правило, если у вас нет web сервера. И наконец порт 113, ответственный за службу IDENTD и использующийся некоторыми протоколами типа IRC, и пр. Замечу, что вам следует использовать пакет `oidentd` если вы делаете трансляцию сетевых адресов для некоторых узлов (хостов) в локальной сети. `oidentd` поддерживает передачу IDENTD запросов в локальной сети.

Если у вас имеется необходимость открыть дополнительные порты, то просто скопируйте одно из правил в цепочке `tcp_packets` и подправьте номера портов в соответствии с вашими требованиями.

7.2.6.4. Цепочка для UDP

Пакеты UDP из цепочки INPUT следуют в цепочку `udp_packets` Как и в случае с TCP пакетами, здесь они проверяются на допустимость по номеру порта назначения. Обратите внимание - мы не проверяем исходящий порт пакета, поскольку об этом заботится механизм определения состояния. Открываются только те порты, которые обслуживаются серверами или демонами на нашем брандмауэре. Пакеты, которые поступают на брандмауэр по уже установленным соединениям (установленным из локальной сети) пропускаются брандмауэром автоматически, поскольку имеют состояние `ESTABLISHED` или `RELATED`.

Как видно из текста сценария, порт 53, на котором "сидит" DNS, для UDP пакетов закрыт, то есть правило, открывающее 53-й порт в сценарии присутствует, но закомментировано. Если вы пожелаете запустить DNS на брандмауэре, то это правило следует раскомментировать.

Я лично разрешаю порт 123, на котором работает NTP (*network time protocol*). Этой службой обычно пользуются для приема точного времени с серверов времени в Интернет. Однако, вероятнее всего, что вы не используете этот протокол, поэтому соответствующее правило в сценарии так же закомментировано.

Порт 2074 используется некоторыми мультимедийными приложениями, подобно *speak freely*, которые используются для передачи голоса в режиме реального времени.

И наконец - ICQ, на порту 4000. Это широко известный протокол, используемый ICQ-приложениями Я полагаю, не следует объяснять вам что это такое.

Кроме того в сценарии приведены еще два правила, которые по-умолчанию закомментированы. Ими можно воспользоваться, если брандмауэр чрезмерно нагружен. Первое - блокирует широковещательные пакеты, поступающие на порты со 135 по 139. Эти порты используются протоколами SMB и NetBIOS от Microsoft. Таким образом данное правило предотвращает переполнение таблицы трассировщика в сетях Microsoft Network. Второе правило блокирует DHCP запросы извне. Это правило определенно имеет смысл если внешняя сеть содержит некоммутируемые сегменты, где IP адреса выделяются клиентам динамически.

Последние два правила не являются обязательными (в тексте сценария они закомментированы). Все пакеты, которые не были отвергнуты или приняты явно, логируются в журнал последним правилом в цепочке INPUT, поэтому, если вас беспокоит проблема "раздувания" системного журнала -- можете раскомментировать эти правила.

7.2.6.5. Цепочка для ICMP

Здесь принимается решение о пропуске ICMP пакетов. Если пакет приходит с eth0 в цепочку INPUT, то далее он перенаправляется в цепочку icmp_packets. В этой цепочке проверяется тип ICMP сообщения. Пропускаются только ICMP Echo Request, TTL equals 0 during transit и TTL equals 0 during reassembly. Все остальные типы ICMP сообщений должны проходить брандмауэр беспрепятственно, поскольку будут иметь состояние RELATED.

Если ICMP пакет приходит в ответ на наш запрос, то он приобретает статус RELATED (связанный с имеющимся соединением). Более подробно состояние пакетов рассматривается в главе Механизм определения состояний

При принятии решения я исхожу из следующих соображений: ICMP Echo Request пакеты посылаются, главным образом, для проверки доступности хоста. Если удалить это правило, то наш брандмауэр не будет "откликаться" в ответ на ICMP Echo Request, что сделает использование утилиты ping и подобных ей, по отношению к брандмауэру, бесполезными.

Time Exceeded (т.е., TTL equals 0 during transit и TTL equals 0 during reassembly). Во время движения пакета по сети, на каждом маршрутизаторе поле TTL, в заголовке пакета, уменьшается на 1. Как только поле TTL станет равным нулю, то маршрутизатором будет послано сообщение Time Exceeded. Например, когда вы выполняете трассировку (*traceroute*) какого либо узла, то поле TTL устанавливается равным 1, на первом же маршрутизаторе оно становится равным нулю и к нам приходит сообщение Time Exceeded, далее, устанавливаем TTL = 2 и второй маршрутизатор передает нам Time Exceeded, и так далее, пока не получим ответ с самого узла.

Список типов ICMP сообщений смотрите в приложении Типы ICMP. Дополнительную информацию по ICMP вы можете получить в следующих документах:

- [The Internet Control Message Protocol](#)
- "RFC 792 - Internet Control Message Protocol - от J. Postel":

Будьте внимательны при блокировании ICMP пакетов, возможно я не прав, блокируя какие-то из них, может оказаться так, что для вас это неприемлемо.

7.2.7. Цепочка INPUT

Цепочка INPUT, как я уже писал, для выполнения основной работы использует другие цепочки, за счет чего снижая нагрузку на сетевой фильтр. Эффект применения такого варианта организации правил лучше заметен на медленных машинах, которые в другом случае начинают "терять" пакеты при высокой нагрузке. Достигается это разбиением набора правил по некоторому признаку и выделение их в отдельные цепочки. Тем самым уменьшается количество правил, которое проходит каждый пакет.

Первым же правилом мы пытаемся отбросить "плохие" пакеты. За дополнительной информацией обращайтесь к приложению Цепочка bad_tcp_packets. В некоторых особенных ситуациях такие пакеты могут считаться допустимыми, но в 99% случаев лучше их "остановить". Поэтому такие пакеты заносятся в системный журнал (логируются) и "сбрасываются".

Далее следует целая группа правил, которая пропускает весь трафик, идущий из доверительной сети, которая включает в себя сетевой адаптер, связанный с локальной сетью и локальный сетевой интерфейс (`lo`) и имеющий исходные адреса нашей локальной сети (включая реальный IP адрес). Эта группа правил стоит первой по той простой причине, что локальная сеть генерирует значительно больше трафика чем трафик из Internet. Поэтому, при построении своих наборов правил, всегда старайтесь учитывать объем трафика, указывая первыми те правила, которые будут обслуживать больший трафик.

Первым в группе, анализирующей трафик идущий с `$INET_INTERFACE`, стоит правило, пропускающее все пакеты со статусом `ESTABLISHED` или `RELATED` (эти пакеты являются частью уже УСТАНОВЛЕННОГО или СВЯЗАННОГО соединения). Это правило эквивалентно правилу, стоящему в цепочке `allowed`. И в некоторой степени является избыточным, поскольку затем цепочка `allowed` вызывается опосредованно через цепочку `tcp_packets`, однако оно несколько разгружает сетевой фильтр, поскольку значительная доля трафика пропускается этим правилом и не проходит всю последовательность до цепочки `allowed`.

После этого производится анализ трафика, идущего из Internet. Все пакеты, приходящие в цепочку `INPUT` с интерфейса `$INET_INTERFACE` распределяются по вложенными цепочками, в зависимости от типа протокола. TCP пакеты передаются в цепочку `tcp_packets`, UDP пакеты отправляются в цепочку `udp_packets` и ICMP перенаправляются в цепочку `icmp_packets`. Как правило, большую часть трафика "съедают" TCP пакеты, потом UDP и меньший объем приходится на долю ICMP, однако в вашем конкретном случае это предположение может оказаться неверным. Очень важно учитывать объем трафика, проходящего через набор правил. Учет объема трафика - абсолютная необходимость. В случае неоптимального распределения правил даже машину класса Pentium III и выше, с сетевой картой 100 Мбит и большим объемом передаваемых данных по сети, довольно легко можно "поставить на колени" сравнительно небольшим объемом правил.

Далее следует весьма специфическое правило (по-умолчанию закомментировано). Дело в том, что клиенты Microsoft Network имеют "дурную привычку" выдавать огромное количество Multicast (групповых) пакетов в диапазоне адресов 224.0.0.0/8. Поэтому можно использовать данное правило для предотвращения "засорения" логов в случае, если с внешней стороны имеется какая либо сеть Microsoft Network. Подобную же проблему решают два последних правила (по-умолчанию закомментированы) в цепочке `udp_packets`, описанные в Цепочка для UDP.

Последним правилом, перед тем как ко всем не принятным явно пакетам в цепочке `INPUT` будет применена политика по-умолчанию, трафик журналируется, на случай необходимости поиска причин возникающих проблем. При этом мы устанавливаем правилу, ограничение на количество логируемых пакетов - не более 3-х в минуту, чтобы предотвратить чрезмерное раздувание журнала и кроме того подобные записи в журнал сопровождаются собственным комментарием (префиксом), чтобы знать откуда появились эти записи.

Все что не было явно пропущено в цепочке `INPUT` будет подвергнуто действию `DROP`, поскольку именно это действие назначено в качестве политики по-умолчанию. Политики по-умолчанию были описаны чуть выше в разделе Установка политик по-умолчанию.

7.2.8. Цепочка FORWARD

Цепочка FORWARD содержит очень небольшое количество правил. Первое правило направляет все TCP пакеты на проверку в цепочку `bad_tcp_packets`, которая используется так же и в цепочке INPUT. Цепочка `bad_tcp_packets` сконструирована таким образом, что может вызываться из других цепочек, невзирая на то, куда направляется пакет. После проверки TCP пакетов, как обычно, мы разрешим движение пакетов из локальной сети без ограничений.

Далее, пропускается весь трафик из локальной сети без ограничений. Естественно, нужно пропустить ответные пакеты в локальную сеть, поэтому следующим правилом мы пропускаем все, что имеет признак `ESTABLISHED` или `RELATED`, т.е. мы пропускаем пакеты по соединению установленному ИЗ локальной сети.

И в заключение заносим в системный журнал информацию о сброшенных пакетах, предваряя их префиксом "`IPT FORWARD packet died:` ", чтобы потом, в случае поиска ошибок, не перепутать их с пакетами, сброшенными в цепочке INPUT.

7.2.9. Цепочка OUTPUT

Как я уже упоминал ранее, в моем случае компьютер используется как брандмауэр и одновременно как рабочая станция. Поэтому я позволяю покидать мой хост всему, что имеет исходный адрес `$LOCALHOST_IP@`, `$LAN_IP@` или `$STATIC_IP`. Сделано это для защиты от трафика, который может сфальсифицироваться моего компьютера, несмотря на то, что я совершенно уверен во всех, кто имеет к нему доступ. И в довершение ко всему, я журналирую "сброшенные" пакеты, на случай поиска ошибок или в целях выявления сфальсифицированных пакетов. Ко всем пакетам, не прошедшим ни одно из правил, применяется политика по-умолчанию - - `DROP`.

7.2.10. Цепочка PREROUTING таблицы nat

В данном сценарии эта цепочка не имеет ни одного правила и единственно, почему я привожу ее описание здесь, это еще раз напомнить, что в данной цепочке выполняется преобразование сетевых адресов (DNAT) перед тем как пакеты попадут в цепочку INPUT или FORWARD.

Еще раз хочу напомнить, что эта цепочка не предназначена ни для какого вида фильтрации, а только для преобразования адресов, поскольку в эту цепочку попадает только первый пакет из потока.

7.2.11. Запуск SNAT и цепочка POSTROUTING

И заключительный раздел - настройка SNAT. По крайней мере для меня. Прежде всего мы добавляем правило в таблицу nat, в цепочку POSTROUTING, которое производит преобразование исходных адресов всех пакетов, исходящих с интерфейса, подключенного к Internet. В сценарии определен ряд переменных, с помощью которых можно использовать для автоматической настройки сценария. Кроме того, использование переменных повышает удобочитаемость скриптов. Ключом - `t` задается имя таблицы, в данном случае `nat`. Команда - `A` добавляет (Add) новое правило в цепочку POSTROUTING, критерий - `o $INET_IFACE` задает исходящий интерфейс, и в конце правила задаем действие над пакетом - - `SNAT`. Таким образом, все пакеты, подошедшие под заданный критерий будут "замаскированы", т.е. будут выглядеть так, как будто они отправлены с нашего узла. Не забудьте указать ключ - - `to-source` с соответствующим IP адресом для исходящих пакетов

В этом сценарии я использую SNAT вместо MASQUERADE по ряду причин. Первая - предполагается, что этот сценарий должен работать на сетевом узле, который имеет постоянный IP адрес. Следующая состоит в том, что SNAT работает быстрее и более эффективно. Конечно, если вы не имеете постоянного IP адреса, то вы должны использовать действие MASQUERADE, которое предоставляет более простой способ трансляции адресов, поскольку оно автоматически определяет IP адрес, присвоенный заданному интерфейсу. Однако, по сравнению с SNAT это действие требует нескольких больших вычислительных ресурсов, хотя и не значительно. Пример работы с MASQUERADE, вы найдете в сценарии [rc.DHCP.firewall.txt](#)

Глава 8. Примеры сценариев

Цель этой главы состоит в том, чтобы дать краткое описание каждого сценария, в этом руководстве. Эти сценарии не совершенны, и они не могут полностью соответствовать вашим нуждам. Это означает, что вы должны сами "подогнать" эти сценарии под себя. Последующая часть руководства призвана облегчить вам эту подгонку.

8.1. Структура файла rc.firewall.txt

Все сценарии, описанные в этом руководстве, имеют определенную структуру. Сделано это для того, чтобы сценарии были максимально похожи друг на друга, облегчая тем самым поиск различий между ними. Эта структура довольно хорошо описывается в этой главе. Здесь я надеюсь дать вам понимание, почему все сценарии были написаны именно так и почему я выбрал именно эту структуру.

Обратите внимание на то, что эта структура может оказаться далеко неоптимальной для ваших сценариев. Эта структура выбрана лишь для лучшего объяснения хода моих мыслей.

8.1.1. Структура

Это - структура, которой следуют все сценарии в этом руководстве. Если вы обнаружите, что это не так, то скорее всего это моя ошибка, если конечно я не объяснил, почему я нарушил эту структуру.

1. *Configuration* - Прежде всего мы должны задать параметры конфигурации, для сценария. Параметры Конфигурации, в большинстве случаев, должны быть описаны первыми в любом сценарии.
 1. *Internet* - Это раздел конфигурации, описывающей подключение к Internet. Этот раздел может быть опущен, если вы не подключены к Интернет. Обратите внимание, что может иметься большее количество подразделов чем, здесь перечислено, но только те, которые описывают наше подключение к Internet.
 1. *DHCP* - Если имеются специфичные для DHCP настройки, то они добавляются здесь.
 2. *PPPoE* - Описываются параметры настройки PPPoE подключения.
 2. *LAN* - Если имеется любая ЛОКАЛЬНАЯ СЕТЬ за брандмауэром, то здесь указываются параметры, имеющие отношение к ней. Наиболее вероятно, что этот раздел будет присутствовать почти всегда.
 3. *DMZ* - Здесь добавляется конфигурация зоны DMZ. В большинстве сценариев этого раздела не будет, т.к. любая нормальная домашняя сеть, или маленькая локальная сеть, не будет иметь ее. (*DMZ* - de-militarized zone. Скорее всего под это понятие автор подвел небольшую подсеть, в которой расположены серверы, например: DNS, MAIL, WEB и т.п, и нет ни одной пользовательской машины. прим. перев.)
 4. *Localhost* - Эти параметры принадлежат нашему брандмауэру (localhost). В вашем случае эти переменные вряд ли изменятся, но, тем не менее, я создал эти переменные. Хотелось бы надеяться, что у вас не будет причин изменять эти переменные.
 5. *iptables* - Этот раздел содержит информацию об iptables. В большинстве сценариев достаточно будет только одной переменной, которая указывает путь к iptables.
 6. *Other* - Здесь располагаются прочие настройки, которые не относятся и к одному из вышеуказанных разделов.
2. *Module loading* - Этот раздел сценариев содержит список модулей. Первая часть должна содержать требуемые модули, в то время как вторая часть должна содержать нетребуемые модули.

bq. Обратить внимание. Некоторые модули, отвечающие за дополнительные возможности, могут быть указаны даже если они не требуются. Обычно, в таких случаях, пример сценария отмечает эту особенность.

 1. *Required modules* - Этот раздел должен содержать модули, необходимые для работы сценария.
 2. *Non-required modules* - Этот раздел содержит модули, которые не требуются для нормальной работы сценария. Все эти модули должны быть закомментированы. Если вам они потребуются, то вы должны просто раскомментировать их.

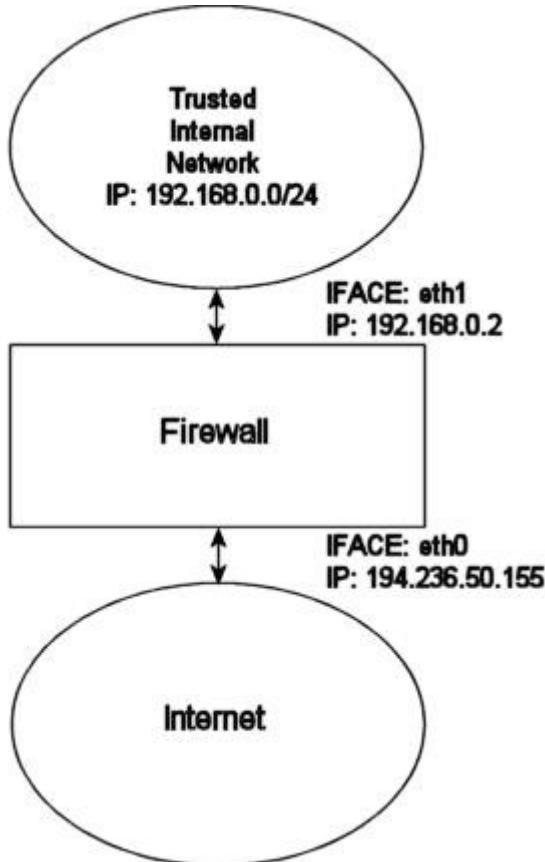
3. *proc configuration* - Этот раздел отвечает за настройку файловой системы `/proc`. Если эти параметры необходимы - они будут перечислены, если нет, то они должны быть закомментированы по-умолчанию, и указаны как не-требуемые. Большинство полезных настроек `/proc` будут перечислены в примерах, но далеко не все.
 1. *Required proc configuration* - Этот раздел должен содержать все требуемые сценарием настройка для `/proc`. Это могут быть настройки для запуска системы защиты, возможно, добавляют специальные возможности для администратора или пользователей.
 2. *Non-required proc configuration* - Этот раздел должен содержать не-требуемые настройки `/proc`, которые могут оказаться полезными в будущем. Все они должны быть закомментированы, так как они фактически не требуются для работы сценария. Этот список будет содержать далеко не все настройки `/proc`.
4. *rules set up* - К этому моменту скрипт, как правило, уже подготовлен к тому, чтобы вставлять наборы правил. Я разбил все правила по таблицам и цепочкам. Любые пользовательские цепочки должны быть созданы прежде, чем мы сможем их использовать. Я указываю цепочки и их наборы правил в том же порядке, в каком они выводятся командой `iptables -L`.
 1. *Filter table* - Прежде всего мы проходим таблицу `filter`. Для начала необходимо установить политику по умолчанию в таблице.
 2. *Set policies* - Назначение политик по-умолчанию для системных цепочек. Обычно я устанавливаю `DROP` для цепочек в таблице `filter`, и буду пропускать потоки, которые идут изнутри. Тем самым мы избавимся от всего, что нам неугодно.
 3. *Create user specified chains* - В этом разделе, создаются все пользовательские цепочки, которые мы будем использовать позже в пределах этой таблицы. Мы не сможем использовать эти цепочки в до тех пор, пока не создадим их.
 4. *Create content in user specified chains* - После создания пользовательских цепочек, мы можем заполнить их правилами. Единственная причина, по которой правила для пользовательских цепочек определяются здесь -- это близость к командам, создающим эти цепочки. Вы же можете размещать правила в другом месте вашего сценария.
 5. *INPUT chain* - В этом разделе добавляются правила для цепочки `INPUT`.
вq. Как уже указывалось, я старался следовать порядку, который получается в выводе команды `iptables -L`. Нет серьезных причин, чтобы соблюдать эту структуру, однако, пробуйте избежать смешивания данных из различных таблиц и цепочек, так как станет намного тяжелее читать такой набор правил и выискивать возможные проблемы.
 6. *FORWARD chain* - Здесь мы добавляем правила в цепочку `FORWARD`
 7. *OUTPUT chain* - амой последней в таблице `filter`, заполняется цепочка `OUTPUT`.
5. *nat table* - После таблицы `filter` мы переходим к таблице `nat`. Сделано это по ряду причин. Прежде всего - не следует запускать механизм NAT на ранней стадии, когда еще возможна передача пакетов без ограничений (то есть, когда NAT уже включена, но нет ни одного правила фильтрации). Также, я рассматриваю таблицу `nat` как своего рода уровень, который находится вне таблицы `filter`. Таблица `filter` является своего рода ядром, в то время как `nat` - оболочка вокруг ядра, а таблица `mangle` может рассматриваться как оболочка вокруг таблицы `nat`. Это может быть не совсем правильно, но и не далеко от действительности.
 1. *Set policies* - Прежде всего мы устанавливаем всю политику по умолчанию в пределах таблицы `nat`. Обычно, я устанавливаю `ACCEPT`. Эта таблица не должна использоваться для фильтрации, и мы не должны здесь "выбрасывать" (`DROP`) пакеты. Есть ряд неприятных побочных эффектов которые имеют место быть в таких случаях из-за наших предположений. Я пропускаю все пакеты в этих цепочках, поскольку не вижу никаких причин не делать этого.
 2. *Create user specified chains* - Здесь создаются все пользовательские цепочки для таблицы `nat`. Обычно у меня их нет, но я добавил этот раздел на всякий случай. Обратите внимание, что пользовательские цепочки должны быть созданы до их фактического использования.
 3. *Create content in user specified chains* - Добавление правил в пользовательские цепочки таблицы `nat`. Принцип размещения правил здесь тот же что и в таблице `filter`. Я добавляю их здесь потому, что не вижу причин выносить их в другое место.

4. *PREROUTING chain* - Цепочка PREROUTING используется для DNAT. В большинстве сценариев DNAT не используется, или по крайней мере закомментирована, чтобы не "открывать ворота" в нашу локальную сеть слишком широко. В некоторых сценариях это правило включено, так как единственная цель этих сценариев состоит в предоставлении услуг, которые без DNAT невозможны.
 5. *POSTROUTING chain* - Цепочка POSTROUTING используется сценариями, которые я написал, так как в большинстве случаев имеется одна или более локальных сетей, которые мы хотим подключить к Интернет через сетевой экран. Главным образом мы будем использовать SNAT, но в некоторых случаях, мы вынуждены будем использовать MASQUERADE.
 6. *OUTPUT chain* - Цепочка OUTPUT используется вообще в любом из сценариев. Но я пока не нашел серьезных оснований для использования этой цепочки. Если вы используете эту цепочку, черкните мне пару строк, и я внесу соответствующие изменения в данное руководство.
6. *mangle table* - Таблица mangle - последняя таблица на пути пакетов. Обычно я не использую эту таблицу вообще, так как обычно не возникает потребностей в чем либо, типа изменения TTL поля или поля TOS и пр. Другими словами, я оставил этот раздел пустым в некоторых сценариях, с несколькими исключениями, где я добавил, несколько примеров использования этой таблицы.
1. *Set policies* - Здесь задается Политика по-умолчанию. Здесь существуют те же ограничения, что и для таблицы nat. Таблица не должна использоваться для фильтрации, и следовательно вы должны избегать этого. Я не устанавливал никакой политики в любом из сценариев для цепочек в таблице mangle, и вам следует поступать так же.
 2. *Create user specified chains* - Создаются пользовательские цепочки. Так как я не использую таблицу mangle в сценариях, я не стал создавать пользовательских цепочек. Однако, этот раздел был добавлен на всякий случай.
 3. *Create content in user specified chains* - Если вы создали какие либо пользовательские цепочки в пределах этой таблицы, вы можете заполнить их правилами здесь.
 4. *PREROUTING* - В этом пункте имеется только упоминание о цепочке.
 5. *INPUT chain* - В этом пункте имеется только упоминание о цепочке.
 6. *FORWARD chain* - В этом пункте имеется только упоминание о цепочке.
 7. *OUTPUT chain* - В этом пункте имеется только упоминание о цепочке.
 8. *POSTROUTING chain* - В этом пункте имеется только упоминание о цепочке.

Надеюсь, что я объяснил достаточно подробно, как каждый сценарий структурирован и почему они структурированы таким способом.

Обратите внимание, что эти описания чрезвычайно кратки, и являются лишь кратким пояснением того, почему сценарии имеют такую структуру. Я не претендую на истину в последней инстанции и не утверждаю, что это - единственный и лучший вариант.

8.2. rc.firewall.txt

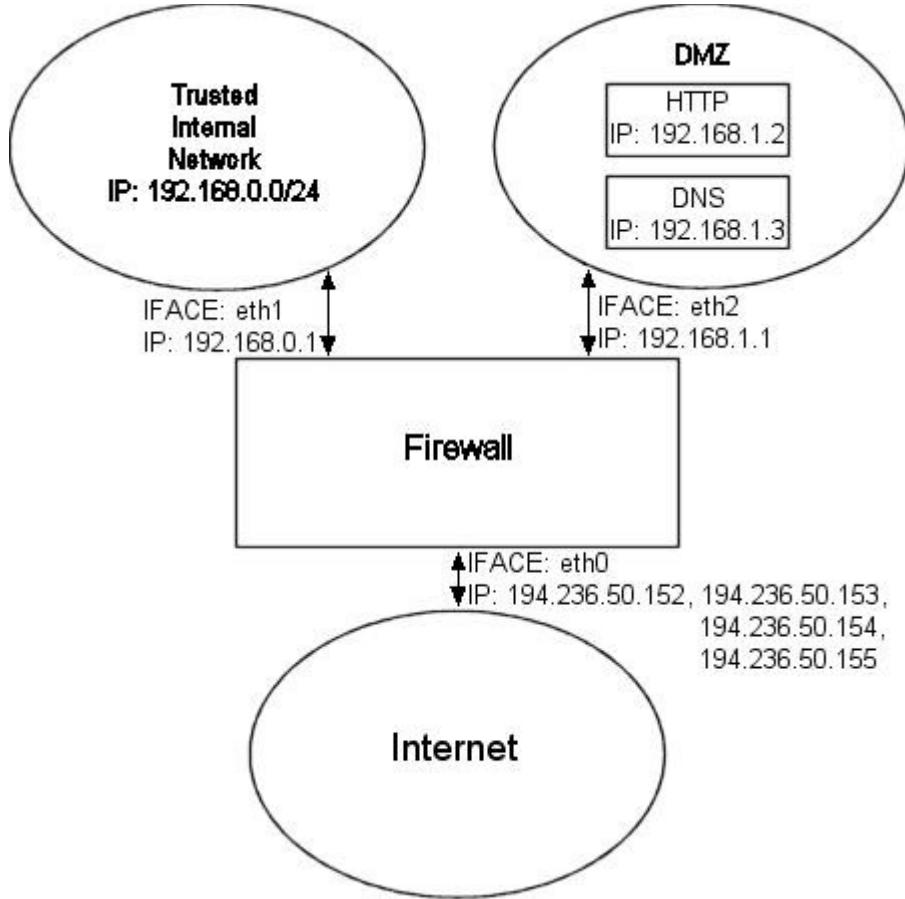


Сценарий [rc.firewall.txt](#) - основное ядро, на котором основывается остальные сценарии. Глава *Файл rc.firewall* достаточно подробно описывает сценарий. Сценарий написан для домашней сети, где вы имеете одну ЛОКАЛЬНУЮ СЕТЬ и одно подключение к Internet. Этот сценарий также исходит из предположения, что вы имеете статический IP адрес, и следовательно не используете DHCP, PPP, SLIP либо какой то другой протокол, который назначает IP динамически. В противном случае возьмите за основу сценарий [rc.DHCP.firewall.txt](#)

Сценарий требует, чтобы следующие опции были скомпилированы либо статически, либо как модули. Без какой либо из них сценарий будет неработоспособен. Кроме того, изменения, которые вы возможно внесете в текст сценария, могут потребовать включения дополнительных возможностей в ваше ядро.

- CONFIG_NETFILTER
- CONFIG_IP_NF_CONNTRACK
- CONFIG_IP_NF_IPTABLES
- CONFIG_IP_NF_MATCH_LIMIT
- CONFIG_IP_NF_MATCH_STATE
- CONFIG_IP_NF_FILTER
- CONFIG_IP_NF_NAT
- CONFIG_IP_NF_TARGET_LOG

8.3. rc.DMZ.firewall.txt



Сценарий [@rc.DMZ.firewall.txt](#) был написан для тех, кто имеет доверительную локальную сеть, одну "Демилитаризованную Зону" и одно подключение к Internet. Для доступа к серверам Демилитаризированной Зоны, в данном случае, извне, используется NAT "один к одному", то есть, Вы должны заставить брандмауэр распознавать пакеты более чем для одного IP адреса.

Сценарий требует, чтобы следующие опции были скомпилированы либо статически, либо как модули. Без какой либо из них сценарий будет неработоспособен

- CONFIG_NETFILTER
- CONFIG_IP_NF_CONNTRACK
- CONFIG_IP_NF_IPTABLES
- CONFIG_IP_NF_MATCH_LIMIT
- CONFIG_IP_NF_MATCH_STATE
- CONFIG_IP_NF_FILTER
- CONFIG_IP_NF_NAT
- CONFIG_IP_NF_TARGET_LOG

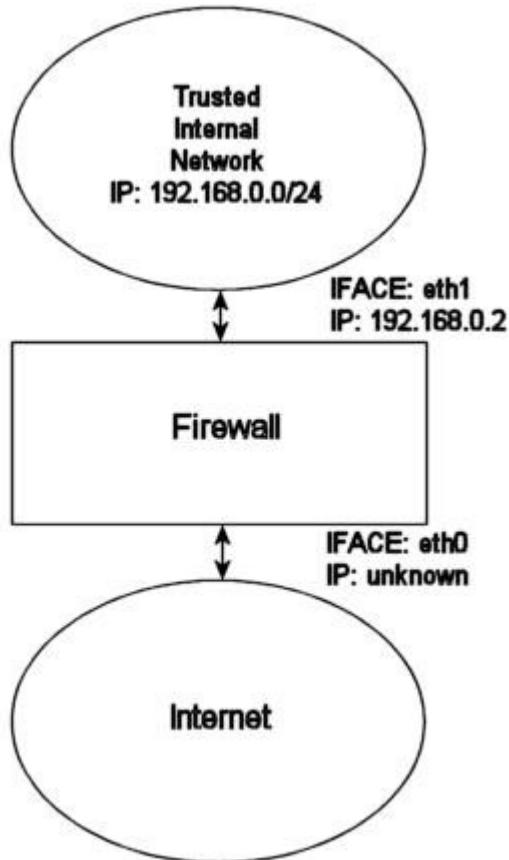
Сценарий работает с двумя внутренними сетями, как это продемонстрировано на рисунке. Одна использует диапазон IP адресов **192.168.0.0/24** и является Доверительной Внутренней Сетью. Другая использует диапазон **192.168.1.0/24** и называется Демилитаризированной Зоной (DMZ), для которой мы будем выполнять преобразование адресов (NAT) "один к одному". Например, если кто-то из Интернет отправит пакет на наш DNS_IP, то мы выполним DNAT для передачи пакета на DNS в DMZ. Если бы DNAT не выполнялся, то DNS не смог бы получить запрос, поскольку он имеет адрес **DMZ_DNS_IP**, а не **DNS_IP**. Трансляция выполняется следующим правилом:

```
$IPTABLES -t nat -A PREROUTING -p TCP -i $INET_IFACE -d $DNS_IP \
--dport 53 -j DNAT --to-destination $DMZ_DNS_IP
```

Для начала напомню, что DNAT может выполняться только в цепочке PREROUTING таблицы nat. Согласно этому правилу, пакет должен приходить по протоколу TCP на \$INET_IFACE с адресатом IP, который соответствует нашему \$DNS_IP, и направлен на порт 53. Если встречен такой пакет, то выполняется подмена адреса назначения, или DNAT. Действию DNAT передается адрес для подмены с помощью ключа --to-destination \$DMZ_DNS_IP. Когда через брандмауэр возвращается пакет ответа, то сетевым кодом ядра отправителя будет автоматически изменен с \$DMZ_DNS_IP на \$DNS_IP, другими словами обратная детрансляция адресов выполняется автоматически и не требует создания дополнительных правил.

Теперь вы уже должны понимать как работает DNAT, чтобы самостоятельно разобраться в тексте сценария без каких либо проблем. Если что-то для вас осталось не ясным и это не было рассмотрено в данном документе, то вы можете сообщить мне об этом - вероятно, это моя ошибка.

8.4. rc.DHCP.firewall.txt



Сценарий [The rc.DHCP.firewall.txt](#) очень похож на оригинал [rc.firewall.txt](#). Однако, этот сценарий больше не использует переменную STATIC_IP, это и является основным отличием от оригинала [rc.firewall.txt](#). Причина в том, что [rc.firewall.txt](#) не будет работать в случае динамического IP адреса. Изменения, по сравнению с оригиналом - минимальны. Этот сценарий будет полезен в случае DHCP, PPP и SLIP подключения к Интернет.

Сценарий требует, чтобы следующие опции были скомпилированы либо статически, либо как модули. Без какой либо из них сценарий будет неработоспособен

- CONFIG_NETFILTER
- CONFIG_IP_NF_CONNTRACK
- CONFIG_IP_NF_IPTABLES
- CONFIG_IP_NF_MATCH_LIMIT
- CONFIG_IP_NF_MATCH_STATE
- CONFIG_IP_NF_FILTER
- CONFIG_IP_NF_NAT
- CONFIG_IP_NF_TARGET_MASQUERADE
- CONFIG_IP_NF_TARGET_LOG

Главное отличие данного скрипта состоит в удалении переменной STATIC_IP и всех ссылок на эту переменную. Вместо нее теперь используется переменная INET_IFACE. Другими словами - `-d $STATIC_IP` заменяется на `-i $INET_IFACE`. Собственно, это все, что нужно изменить в действительности. (Хочется отметить, что в данном случае под STATIC_IP автор понимает переменную INET_IP прим. перев.)

Мы больше не можем устанавливать правила в цепочке INPUT подобных этому: `--in-interface $LAN_IFACE --dst $INET_IP`. Это в свою очередь вынуждает нас строить правила основываясь только на сетевом интерфейсе. Например, пусть на брандмауэре запущен HTTP сервер. Если мы приходим на главную страницу, содержащую статическую ссылку обратно на этот же сервер, который работает под динамическим адресом, то мы можем "огрести" немало проблем. Хост, который проходит через NAT, запросит через DNS IP адрес HTTP сервера, после чего попробует получить доступ к этому IP. Если брандмауэр производит фильтрацию по интерфейсу и IP адресу, то хост не сможет получить ответ, поскольку цепочка INPUT отфильтрует такой запрос. Это так же справедливо и для некоторых случаев когда мы имеем статический IP адрес, но тогда это можно обойти, используя правила, которые проверяют пакеты, приходящие с LAN интерфейса на наш INET_IP и выполнять ACCEPT для них.

После всего вышесказанного, не такой уж плохой может показаться мысль о создании сценария, который бы обрабатывал динамический IP. Например, можно было бы написать скрипт, который получает IP адрес через `ifconfig` и подставляет его в текст сценария (где определяется соответствующая переменная), который "поднимает" соединение с Интернет. Замечательный сайт linuxguruz.org имеет огромную коллекцию скриптов, доступных для скачивания. Ссылку на linuxguruz.org вы найдете в приложении *Ссылки на другие ресурсы*.

Этот сценарий менее безопасен чем [rc.firewall.txt](#). Я настоятельно рекомендую вам использовать сценарий [rc.firewall.txt](#), если это возможно, так как [rc.DHCP.firewall.txt](#) более открыт для нападений извне.

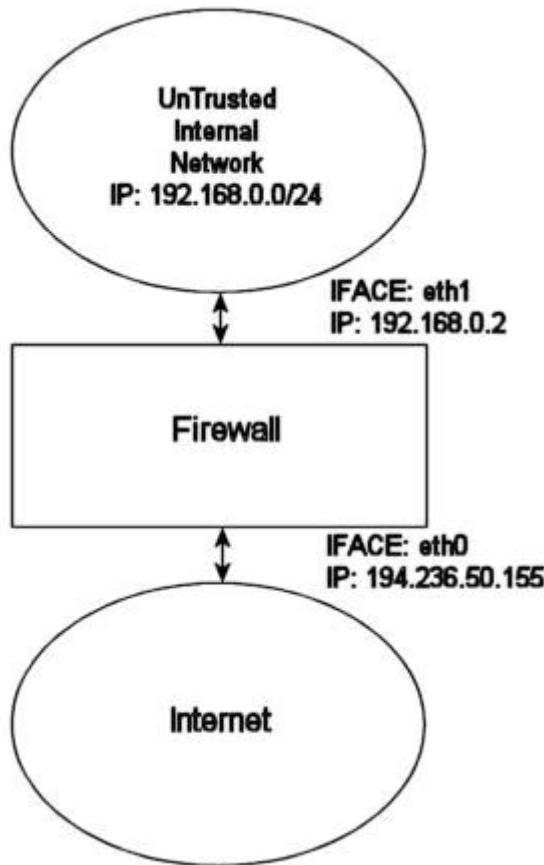
Также, можно добавить в ваши сценарии что нибудь вроде этого:

```
INET_IP=`ifconfig $INET_IFACE | grep inet | cut -d : -f 2 | \
cut -d ' ' -f 1`
```

Выше приведенная команда получает динамический IP от интерфейса. Более совершенные методы получения IP адреса вы найдете в сценарии [retreiveip.txt](#). Однако у такого подхода есть серьезные недостатки, которые описаны ниже.

1. Если скрипт запускается из другого сценария, который в свою очередь запускается демоном PPP, то это может привести к "зависанию" всех, уже установленных соединений, из-за правил, которые отбраковывают пакеты со статусом NEW и со сброшенным битом SYN. (смотри раздел *Пакеты со статусом NEW и со сброшенным битом SYN*). Проблему конечно можно разрешить удалением этих правил, но такое решение довольно сомнительно с точки зрения безопасности.
2. Предположим, что у вас есть набор статических правил, довольно грубо будет постоянно стирать и добавлять правила, к тому же рискуя повредить существующие.
3. Это может привести к излишним усложнениям, что в свою очередь, влечет ослабление защиты. Чем проще скрипт, тем проще его сопровождать.

8.5. rc.UTIN.firewall.txt



Сценарий [rc.UTIN.firewall.txt](#), в отличие от других сценариев, блокирует LAN, которая находится за брандмауэром. Мы доверяем внутренним пользователям не больше чем пользователям из Internet. Другими словами, мы не доверяем никому, ни в Internet, ни в локальной сети, с которыми мы связаны. Поэтому доступ к Internet ограничивается только протоколами POP3, HTTP и FTP.

Этот сценарий следует золотому правилу - "не доверяй никому, даже собственным служащим". Это грустно но факт - большая часть атак и взломов, которым подвергается компания, производится служащими компаний из локальных сетей. Этот сценарий, надеюсь, даст некоторые сведения, которые помогут вам усилить вашу межсетевую защиту. Он мало отличается от оригинала [rc.firewall.txt](#), но содержит подсказки о том, что мы обычно пропускаем.

Сценарий требует, чтобы следующие опции были скомпилированы либо статически, либо как модули. Без какой либо из них сценарий будет неработоспособен:

- CONFIG_NETFILTER
- CONFIG_IP_NF_CONNTRACK
- CONFIG_IP_NF_IPTABLES
- CONFIG_IP_NF_MATCH_LIMIT
- CONFIG_IP_NF_MATCH_STATE
- CONFIG_IP_NF_FILTER
- CONFIG_IP_NF_NAT
- CONFIG_IP_NF_TARGET_LOG

8.6. rc.test-iptables.txt

Сценарий [rc.test-iptables.txt](#) предназначен для проверки различных цепочек но может потребовать дополнительных настроек, в зависимости от вашей конфигурации, например, включения `ip_forwarding` или настройки `masquerading` и т.п. Тем не менее в большинстве случаев с базовыми настройками, когда настроены основные таблицы, этот сценарий будет работоспособен. В действительности, в этом сценарии производится установка действий LOG на ping-запросы и ping-ответы. Таким способом появляется возможность зафиксировать в системном журнале какие цепочки проходились и в каком порядке. Запустите сценарий и затем выполните следующие команды:

```
ping -c 1 host.on.the.internet
```

И во время исполнения первой команды выполните `tail -n 0 -f /var/log/messages`. Теперь вы должны ясно видеть все используемые цепочки и порядок их прохождения.

Note

Этот сценарий был написан исключительно в демонстрационных целях. Другими словами, не следует иметь правила для журналирования подобно этим, которые регистрируют все пакеты без ограничений. В противном случае вы рискуете стать легкой добычей для злоумышленника, который может засыпать вас пакетами, "раздуть" ваш лог, что может вызвать "Отказ в обслуживании", а после этого перейти к реальному взлому вашей системы не боясь быть обнаруженым, поскольку не сможет быть зарегистрирован системой.

8.7. rc.flush-iptables.txt

Сценарий [rc.flush-iptables.txt](#) в действительности не имеет самостоятельной ценности поскольку он сбрасывает все ваши таблицы и цепочки. В начале сценария, устанавливаются политики по-умолчанию ACCEPT для цепочек INPUT, OUTPUT и FORWARD в таблице filter. После этого сбрасываются в заданную по-умолчанию политики для цепочек PREROUTING, POSTROUTING и OUTPUT таблицы nat. Эти действия выполняются первыми, чтобы не возникало проблем с закрытыми соединениями и блокируемыми пакетами. Фактически, этот сценарий может использоваться для подготовки брандмауэра к настройке и при отладке ваших сценариев, поэтому здесь мы заботимся только об очистке набора правил и установке политик по-умолчанию.

Когда выполнена установка политик по-умолчанию, мы переходим к очистке содержимого цепочек в таблицах filter и nat, а затем производится удаление всех, определенных пользователем, цепочек. После этого работа скрипта завершается. Если вы используете таблицу mangle, то вы должны будете добавить в сценарий соответствующие строки для обработки этой таблицы.

В заключение пару слов. Очень многие спрашивают меня, а почему бы не поместить вызов этого сценария в `rc.firewall`, написав что-нибудь типа `rc.firewall start` для запуска скрипта. Я не сделал этого до сих пор, потому что считаю, что учебный материал должен нести в себе основные идеи и не должен быть перегружен разнообразными сценариями со странным синтаксисом. Добавление специфичного синтаксиса делает сценарии менее читабельными, а сам учебный материал более сложным в понимании, поэтому данное руководство остается таким, каково оно есть, и продолжит оставаться таким.

8.8. Limit-match.txt

Сценарий [limit-match.txt](#) написан с целью продемонстрировать работу с критерием `limit`. Запустите этот скрипт и попробуйте отправлять на этот хост ping-пакеты с различными интервалами.

8.9. Pid-owner.txt

Сценарий [pid-owner.txt](#) демонстрирует использование критерия `--pid-owner`. Фактически, этот сценарий ничего не блокирует, поэтому, чтобы увидеть его действие, вам потребуется воспользоваться командой `iptables -L -v`.

8.10. Sid-owner.txt

Сценарий [sid-owner.txt](#) демонстрирует использование критерия `--sid-owner`. Фактически, этот сценарий ничего не блокирует, поэтому, чтобы увидеть его действие, вам потребуется воспользоваться командой `iptables -L -v`.

8.11. Ttl-inc.txt

Небольшой пример [ttl-inc.txt](#), демонстрирующий как можно сделать брандмауэр/роутер "невидимым" для трассировщиков, осложняя тем самым работу атакующего.

8.12. Iptables-save ruleset

Небольшой пример [iptsave-saved.txt](#), о котором говорилось в главе Сохранение и восстановление больших наборов правил, иллюстрирующий работу команды iptables-save. Не является исполняемым сценарием и предназначен лишь для демонстрации результата работы iptables-save.

Приложение А. Детальное описание специальных команд

A.1. Вывод списка правил

Чтобы вывести список правил нужно выполнить команду `iptables` с ключом `L`, который кратко был описан ранее в главе *Как строить правила*. Выглядит это примерно так:

```
iptables -L
```

Эта команда выведет на экран список правил в удобочитаемом виде. Номера портов будут преобразованы в имена служб в соответствии с файлом `/etc/services`, IP адреса будут преобразованы в имена хостов через разрешение имен в службе DNS. С разрешением (resolving) имен могут возникнуть некоторые проблемы, например, имея сеть `192.168.0.0/16` служба DNS не сможет определить имя хоста с адресом `192.168.1.1`, в результате произойдет подвисание команды. Чтобы обойти эту проблему следует выполнить вывод списка правил с дополнительным ключом:

```
iptables -L -n
```

Чтобы вывести дополнительную информацию о цепочках и правилах, выполните

```
iptables -L -n -v
```

Не забывайте о ключе `-t`, который может быть использован для просмотра таблиц `nat` и `mangle`, например:

```
iptables -L -t nat
```

В файловой системе `/proc` имеется ряд файлов, которые содержат достаточно интересную для нас информацию. Например, допустим нам захотелось просмотреть список соединений в таблице `conntrack`. Это основная таблица, которая содержит список трассируемых соединений и в каком состоянии каждое из них находится. Для просмотра таблицы выполните команду:

```
cat /proc/net/ip_conntrack | less
```

A.2. Изменение и очистка ваших таблиц

По мере того как вы продолжите углубляться в исследование `iptables`, перед вами все актуальнее будет вставать вопрос об удалении отдельных правил из цепочек без необходимости перезагрузки машины. Сейчас я попробую на него ответить. Если вы по ошибке добавили какое либо правило, то вам нужно только заменить команду `-A` на команду `-D` в строке правила. `iptables` найдет заданное правило и удалит его. Если имеется несколько правил, которые выглядят как заданный шаблон для удаления, то будет стерто первое из найденных правил. Если такой порядок вещей вас не устраивает, то команде `-D`, в качестве параметра, можно передать номер удаляемой строки, например, команда `iptables -D INPUT 10` сотрет десятое правило в цепочке `INPUT`. (Чтобы узнать номер правила, подайте команду `iptables -L НАЗВАНИЕ_ЦЕПОЧКИ --line-numbers`, тогда правила будут выводиться со своими номерами прим. перев.)

Для удаления содержимого целой цепочки используйте команду `-F`. Например: `iptables -F INPUT` - сотрет все правила в цепочке `INPUT`, однако эта команда не изменяет политики цепочки по-умолчанию, так что если она установлена как `DROP`, то будет блокироваться все, что попадает в цепочку `INPUT`. Чтобы сбросить политику по-умолчанию, нужно просто установить ее в первоначальное состояние, например `iptables -P INPUT ACCEPT`. (И еще: если таблица не указана явно ключом `-t` (`--table`), то очистка цепочек производится ТОЛЬКО в таблице `filter`, прим. перев.)

Мною был написан небольшой сценарий (описанный несколько выше), который производит очистку всех таблиц и цепочек, и переустанавливает политики цепочек в iptables. Запомните, что при использовании таблицы `mangle` вам необходимо внести дополнения в этот сценарий, поскольку он ее не обрабатывает.

Приложение В. Общие проблемы и вопросы

B.1. Проблемы загрузки модулей

Вы можете столкнуться с некоторыми проблемами при попытке загрузить тот или иной модуль.

Например, может быть выдано сообщение об отсутствии запрашиваемого модуля

```
insmod: iptable_filter: no module by that name found
```

Пока еще нет причин для беспокойства. Вполне возможно, что запрашиваемый модуль (или модули) был связан с ядром статически. Это первое, что вы должны проверить. В примере, приведенном выше, произошла ошибка при загрузке таблицы `filter`. Чтобы проверить наличие этой таблицы просто запустите команду:

```
iptables -t filter -L
```

Если все нормально, то эта команда выведет список всех цепочек из таблицы `filter`. Вывод должен выглядеть примерно так:

```
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Если таблица `filter` отсутствует, то вывод будет примерно следующим

```
iptables v1.2.5: can't initialize iptables table `filter': Table \
                  does not exist (do you need to insmod?)
Perhaps iptables or your kernel needs to be upgraded.
```

Это уже серьезнее, так как это сообщение указывает на то, что либо вы забыли установить модули, либо вы забыли выполнить `depmod -a`, либо вы вообще не скомпилировали необходимые модули. Для решения первой проблемы запустите команду `make modules_install` в каталоге с исходными текстами ядра. Вторая проблема решается запуском команды `depmod -a`. Разрешение третьей проблемы уже выходит за рамки данного руководства, и в этом случае рекомендую посетить домашнюю страницу [The Linux Documentation Project](#). (Взгляните еще раз в начало документа, где описывается процесс установки `iptables`. прим. перев.)

Другие ошибки, которые вы можете получить при запуске `iptables`:

```
iptables: No chain/target/match by that name
```

Эта ошибка сообщает, что нет такой цепочки, действия или критерия. Это может зависеть от огромного числа факторов, наиболее вероятно, что вы пытаетесь использовать несуществующую (или еще не определенную) цепочку, несуществующее действие или критерий. Либо потому, что не загружен необходимый модуль.

B.2. Пакеты со статусом NEW и со сброшенным битом SYN

Это свойство iptables недостаточно хорошо задокументировано, а поэтому многие могут уделить ему недостаточное внимание (включая и меня). Если вы используете правила, определяющие статус пакета NEW, но не проверяете состояние бита SYN, то пакеты со сброшенным битом SYN смогут "просочиться" через вашу защиту. Хотя, в случае, когда мы используем несколько брандмауэров, такой пакет может оказаться частью ESTABLISHED соединения, установленного через другой брандмауэр. Пропуская подобные пакеты, мы делаем возможным совместную работу двух или более брандмауэров, при этом мы можем любой из них остановить, не боясь разорвать установленные соединения, поскольку функции по передаче данных тут же возьмет на себя другой брандмауэр. Однако это позволит устанавливать практически любое TCP соединение. Во избежание этого следует добавить следующие правила в цепочки INPUT, OUTPUT и FORWARD:

```
$IPTABLES -A INPUT -p tcp ! --syn -m state --state NEW -j LOG \
--log-prefix "New not syn:"
$IPTABLES -A INPUT -p tcp ! --syn -m state --state NEW -j DROP
```

Вышеприведенные правила позаботятся об этой проблеме. Будьте чрезвычайно внимательны при построении правил принимающих решение на основе статуса пакета.

Обратите внимание, что имеются некоторые неприятности с вышеприведенными правилами и плохой реализацией TCP/IP от Microsoft. Дело в том, что при некоторых условиях, пакеты, сгенерированные программами от Microsoft маркируются как NEW и согласно этим правилам будут сброшены. Это, однако, не приводит к разрушению соединений, насколько я знаю. Происходит это потому, что, когда соединение закрывается, и посыпается завершающий пакет FIN/ACK, то netfilter закрывает это соединение и удаляет его из таблицы conntrack. В этот момент, дефектный код Microsoft посыпает другой пакет, которому присваивается статус NEW, но в этом пакете не установлен бит SYN и, следовательно, соответствует вышеупомянутым правилам. Короче говоря - особо не переживайте по поводу этих правил. В случае чего - вы сможете просмотреть системный журнал, куда логируются отбрасываемые пакеты (см. правила выше) и разобраться с ними.

Имеется еще одна известная проблема с этими правилами. Если кто-то в настоящее время связан с брандмауэром, например из LAN, и активирует PPP, то в этом случае соединение будет уничтожено. Это происходит в момент, когда загружаются или выгружаются conntrack и nat модули. Другой способ получить эту проблему состоит в том, чтобы выполнить сценарий [rc.firewall.txt](#) из сеанса telnet с другого компьютера. Для этого вы соединяетесь по telnet с брандмауэром. Запускаете [rc.firewall.txt](#), в процессе исполнения которого, запускаются модули трассировки подключений, грузятся правила "NEW not SYN". Когда клиент telnet или daemon пробуют послать что нибудь, то это подключение будет распознано трассировочным кодом как NEW, но пакеты не имеют установленного бита SYN, так как они, фактически, являются частью уже установленного соединения. Следовательно, пакет будет соответствовать правилам в результате чего будет зажурнилирован и сброшен.

B.3. SYN/ACK - пакеты и пакеты со статусом NEW

Существует одна из разновидностей спуфинг-атак (от англ. spoofing - мистификация, подмена. прим. перев.), которая называется "Предсказание номера TCP-последовательности" (Sequence Number Prediction). Смысл атак такого рода заключается в использовании чужого IP-адреса для нападения на какой либо узел сети.

Для рассмотрения типичной Sequence Number Prediction атаки обозначим через [A] - атакующий хост, [V] - атакуемый хост, [O] - третий хост, чей IP-адрес используется атакующим.

1. Хост [A] отправляет SYN-пакет (запрос на соединение прим. перев.) хосту [V] с обратным IP-адресом хоста [O].
2. Хост [V] отвечает хосту [O] пакетом SYN/ACK.

3. Теперь, по логике вещей, хост [O] должен разорвать соединение пакетом RST, поскольку он не посыпал запрос на соединение (пакет SYN) и попытка атаки провалится, однако, допустим, что хост [O] не ответил (оказался выключененным, перегружен работой или находится за брандмауэром, который не пропустил пакет SYN/ACK).
4. Если хост [O] не отправил пакет RST, прервав таким образом начавшуюся атаку, то атакующий хост [A] получает возможность взаимодействия с хостом [V], выдавая себя за [O].

Не передав RST-пакет, мы, тем самым, способствуем выполнению атаки на хост [V], которая может быть инкриминирована нам самим. Общепринятой считается необходимость отправления пакета RST в подобных случаях (RST в ответ на незапрошенный SYN/ACK). Если в вашем брандмауэре используются правила, фильтрующие пакеты со статусом NEW и сброшенным битом SYN, то SYN/ACK-пакеты будут "сбрасываться" этими правилами. Поэтому, следующее правило необходимо вставить в цепочку `bad_tcp_packets` первым:

```
iptables -A bad_tcp_packets -p tcp --tcp-flags SYN,ACK SYN,ACK \
-m state --state NEW -j REJECT --reject-with tcp-reset
```

В большинстве случаев подобные правила обеспечивают достаточный уровень безопасности для хоста [O] и риск от их использования относительно невелик. Исключение составляют случаи использования серии брандмауэров. В этом случае некоторые соединения могут оказаться заблокированными, даже если они вполне законны. Эти правила, ко всему прочему, допускают некоторые виды сканирования портов, но не более того.

B.4. Поставщики услуг Internet, использующие зарезервированные IP-адреса

Я добавил этот раздел чтобы предупредить вас о туповатых провайдерах (*Internet Service Providers*), которые назначают IP адреса, отведенные IANA для локальных сетей. Например, Swedish Internet Service Provider и телефонная монополия Telia используют такие адреса для своих серверов DNS (диапазон 10.x.x.x). Проблема, с которой вы будете наиболее вероятно сталкиваться, состоит в том, что мы, в своих сценариях, блокируем подключения с любых IP в диапазоне 10.x.x.x, из-за возможности фальсификации пакетов. Если вы столкнетесь с такой ситуацией, то наверное вам придется снять часть правил. Или установить правила, пропускающие траффик с этих серверов, ранее цепочки INPUT, например так:

```
/usr/local/sbin/iptables -t nat -I PREROUTING -i eth1 -s 10.0.0.1/32 -j ACCEPT
```

Хотелось бы напомнить подобным провайдерам, что эти диапазоны адресов не предназначены для использования в Интернет. Для корпоративных сетей - пожалуйста, для ваших собственных домашних сетей - прекрасно! Но вы не должны вынуждать нас "открываться" по вашей прихоти.

B.5. Как разрешить прохождение DHCP запросов через iptables

В действительности, эта задача достаточно проста, если вам известны принципы работы протокола DHCP. Прежде всего необходимо знать, что DHCP работает по протоколу UDP. Следовательно, протокол является первым критерием. Далее, необходимо уточнить интерфейс, например, если DHCP запросы идут через `$LAN_IFACE`, то движение запросов DHCP следует разрешить только через этот интерфейс. И наконец, чтобы сделать правило более определенным, следует уточнить порты. DHCP использует порты 67 и 68. Таким образом, искомое правило может выглядеть следующим образом:

```
$IPTABLES -I INPUT -i $LAN_IFACE -p udp --dport 67:68 --sport 67:68 -j ACCEPT
```

Обратите внимание, это правило пропускает весь трафик по протоколу UDP через порты 67 и 68, однако это не должно вас особенно смущать, поскольку оно разрешает лишь движение запросов от узлов сети, пытающихся установить соединение с портами 67 и 68. Этого правила вполне достаточно, чтобы позволить выполнение DHCP запросов и при этом не слишком широко "открыть ворота". Если вас очень беспокоит проблема безопасности, то вы вполне можете ужесточить это правило.

B.6. Проблемы mIRC DCC

mIRC использует специфичные настройки, которые позволяют соединяться через брандмауэр и обрабатывать DCC соединения должным образом. Если эти настройки используются совместно с iptables, точнее с модулями `ip_conntrack_irc` и `ip_nat_irc`, то эта связка просто не будет работать. Проблема заключается в том, что mIRC автоматически выполняет трансляцию сетевых адресов (NAT) внутри пакетов. В результате, когда пакет попадает в iptables, она просто не знает, что с ним делать. mIRC не ожидает, что брандмауэр будет настолько "умным", чтобы корректно обрабатывать IRC, и поэтому самостоятельно запрашивает свой IP у сервера и затем подставляет его, при передаче DCC запроса.

Включение опции "I am behind a firewall" ("Я за брандмауэром") и использование модулей `ip_conntrack_irc` и `ip_nat_irc` приводят к тому, что `netfilter` пишет в системный журнал сообщение "Forged DCC send packet".

У этой проблемы есть простое решение - отключите эту опцию в mIRC и позвольте iptables выполнять всю работу.

Приложение C. Типы ICMP

Это полный список типов ICMP сообщений: **Таблица C-1. ICMP types** ТИП КОД Описание Запрос Ошибка

- 0 0 Echo Reply x
- 3 0 Network Unreachable x
- 3 1 Host Unreachable x
- 3 2 Protocol Unreachable x
- 3 3 Port Unreachable x
- 3 4 Fragmentation needed but no frag. bit set x
- 3 5 Source routing failed x
- 3 6 Destination network unknown x
- 3 7 Destination host unknown x
- 3 8 Source host isolated (obsolete) x
- 3 9 Destination network administratively prohibited x
- 3 10 Destination host administratively prohibited x
- 3 11 Network unreachable for TOS x
- 3 12 Host unreachable for TOS x
- 3 13 Communication administratively prohibited by filtering x
- 3 14 Host precedence violation x
- 3 15 Precedence cutoff in effect x
- 4 0 Source quench
- 5 0 Redirect for network
- 5 1 Redirect for host
- 5 2 Redirect for TOS and network

5 3 Redirect for TOS and host
8 0 Echo request x
9 0 Router advertisement
10 0 Route solicitation
11 0 TTL equals 0 during transit x
11 1 TTL equals 0 during reassembly x
12 0 IP header bad (catchall error) x
12 1 Required options missing x
13 0 Timestamp request (obsolete) x
14 Timestamp reply (obsolete) x
15 0 Information request (obsolete) x
16 0 Information reply (obsolete) x
17 0 Address mask request x
18 0 Address mask reply x

Приложение D. Ссылки на другие ресурсы

Здесь приведен список ссылок, где вы сможете получить дополнительную информацию :

- [ip-sysctl.txt](#) - из документации к ядру 2.4.14. Маленький, но хороший справочник по организации сетевого кода ядра.
- [ip_dynaddr.txt](#) - из документации к ядру 2.4.14. Маленький справочник по параметрам настройки `ip_dynaddr`, доступным через `sysctl` и файловую систему `/proc`.
- [iptables.8](#) - Маны для iptables 1.2.4 в формате HTML Прекрасное руководство для создания правил в iptables. Всегда полезно иметь под рукой.
- [The Internet Control Message Protocol](#) - Хороший и подробный документ, описывающий протокол ICMP. Написан Ральфом Уолденом (Ralph Walden).
- [RFC 792 - Internet Control Message Protocol](#) - Официальный источник информации по протоколу ICMP. Содержит всю техническую информацию о протоколе ICMP, которая только может потребоваться. Автор J. Postel.
- [RFC 793 - Transmission Control Protocol](#) - Этот документ описывает стандарт протокола TCP. Документ чрезвычайно насыщен техническими подробностями, однако всякий, желающий понять работу протокола TCP во всех деталях, должен прочитать этот документ. Автор J. Postel.
- <http://www.netfilter.org/> - Официальный сайт netfilter и iptables. Необходим для всех желающих установить iptables и netfilter в linux.
- [Firewall rules table](#) - Небольшой файл в формате PDF, любезно предоставленный Стюартом Кларком (Stuart Clark), который представляет из себя набор бланков для ведения отчетности по правилам, используемым на брандмауэре.
- [/etc/protocols](#) - Пример файла `protocols`, полученный в дистрибутиве Slackware. Может служить справочником по номерам протоколов, таких как IP, ICMP или TCP.
- [/etc/services](#) - Пример файла `services`, полученный в дистрибутиве Slackware. Чрезвычайно полезен для просмотра, чтобы увидеть какие протоколы с какими портами работают.
- [Internet Engineering Task Force](#) - Одна из самых больших групп, которые занимаются установлением и поддержкой стандартов Internet. Поддерживает свой репозиторий RFC. Включает в себя как крупные компании, так и отдельные лица, с целью обеспечения межоперабельности Интернета.
- [Linux Advanced Routing and Traffic Control HOW-TO](#) - Один из лучших документов, касающихся роутинга. Поддерживается сайтом Бертом Хубертом (Bert Hubert).
- [Paksecured Linux Kernel patches](#) - На сайте вы найдете все "заплаты" к ядру, написанные Matthew G. Marsh. Среди всего прочего, здесь вы найдете "заплату" FTOS.

- [ULOGD project page](#) - Домашняя страница проекта ULOGD.
- [The Linux Documentation Project](#) один из лучших сайтов, содержащих документацию. Здесь вы найдете огромное количество документов по Linux-тематике.
- <http://www.netfilter.org/documentation/index.html#FAQ> - Официальный FAQ (Frequently Asked Questions) по netfilter .
- <http://www.netfilter.org/documentation/HOWTO/packet-filtering-HOWTO.html> - Rusty Russells Unreliable Guide to packet filtering. Прекрасная документация по основам фильтрации пакетов с помощью iptables, написанная одним из разработчиков iptables и netfilter.
- <http://www.netfilter.org/documentation/HOWTO/NAT-HOWTO.html> - Rusty Russells Unreliable Guide to Network Address Translation. Замечательная документация по Network Address Translation в iptables и netfilter, написанная одним из основных разработчиков Расти Расселом (Rusty Russell).
- <http://netfilter.org/documentation/HOWTO/netfilter-hacking-HOWTO.html> - Rusty Russells Unreliable Netfilter Hacking HOWTO. Один из немногих документов по созданию кода для работы с netfilter и iptables. Так же написан Расти Расселом (Rusty Russell).
- <http://www.linuxguruz.org/iptables/> - Содержит множество ссылок в Интернет по тематике. Имеется список сценариев iptables для различных применений.
- <http://www.faqs.org/docs/gazette/firewalls.html> - Отличное обсуждение по автоматизации работы iptables, например: как, внесением незначительных изменений, заставить ваш компьютер автоматически добавлять "неугодные" сайты в специальный список (banlist) в iptables.
- <http://www.rigacci.org/wiki/lib/exe/fetch.php/doc/appunti/linux/sa/iptables/contrack.html> - Прекрасное описание модулей трассировщика соединений. Если вам интересна тема трассировки соединений, то вам следует это прочитать.
- <http://www.docum.org> - Один из немногих сайтов, который содержит информацию о командах Linux CBQ, tc и ip. Поддерживает сайт - Stef Coene.
- <http://lists.samba.org/mailman/listinfo/netfilter> - Официальный список адресов (mailing-list) по netfilter. Чрезвычайно полезен для разрешения вопросов по iptables и netfilter.

И конечно же, исходный код iptables, документация и люди, которые помогали мне.

Приложение Е. Благодарности

Автор: Oskar Andreasson

blueflux@kofein.net

Copyright (C) 2001-2003 Oskar Andreasson

Перевод: Андрей Киселев (kis_an@mail.ru)

Перекодировка mp3 тегов

Linux уже давно успешно использует универсальную многобайтовую кодировку UTF-8. Засилье операционной системы Windows во многих случаях ломает стандарты. Как пример - всепроникающая кодировка CP1251, заполонившая сайты настолько, что веб-мастера попросту забывают указывать кодировку страниц своих сайтов.

Аналогична ситуация и с тегами mp3, где властвует CP1251.

К счастью, в версии ID3v2.4 появилась возможность хранения тегов в формате UTF-8.

[Calculate Linux Desktop](#) включает пакет *media-libs/mutagen* для работы с тегами. Для перекодирования тегов вашей коллекции mp3-файлов из кодировки CP1251 в UTF-8 перейдите в директорию с музыкальной коллекцией и выполните:

```
find -iname '*.mp3' -print0 | xargs -0 mid3iconv -eCP1251 --remove-v1
```

Флаг "--remove-v1" удаляет записи из первой версии тегов.

Справка по Portage

Portage ([sys-apps/portage](#)) - это система управления пакетами в Gentoo.

- [color.map](#) - пользовательские настройки цвета в Portage
- [dispatch-conf](#) - безопасное обновление конфигурационных файлов после установки новых пакетов
- [ebuild \(1\).html](#) - низкоуровневый интерфейс системы портежей
- [ebuild \(5\).html](#) - внутренний формат, переменные и функции в сценарии ebuild
- [egencache](#) - кэширование метаданных репозитариев ебилдов
- [emaint](#) - проверка состояния системы и обеспечение ее работоспособности
- [emerge](#) - консольный интерфейс Portage
- [env-update](#) - автоматическое обновление настроек окружения
- [etc-update](#) - обработка изменений конфигурационных файлов
- [make.conf](#) - пользовательские настройки Portage
- [portage](#) - главный инструмент, пакетный менеджер Gentoo
- [quickpkg](#) - создание пакетов для дерева портежей
- [repoman](#) - программа Gentoo, обеспечивающая минимальный уровень качества пакетов, добавляемых к дереву портежей
- [xpak](#) - формат данных XPAK, используемый в бинарных пакетах Portage

COLOR.MAP

НАЗВАНИЕ

color.map - пользовательские настройки цвета в Portage

ПАРАМЕТРЫ

`/etc/portage/color.map`

ОПИСАНИЕ

Указанный файл содержит переменные, определяющие классы цвета, которые использует Portage. Проверяя настройки цвета, Portage в первую очередь обращается к нему. Если тот или иной класс цвета не определен в `/etc/portage/color.map`, Portage использует внутренние значения, принятые по умолчанию.

СИНТАКСИС

ПЕРЕМЕННАЯ = [*атрибуты или коды ansi, через пробел*]

АТРИБУТ = [*атрибуты или коды ansi, через пробел*]

ПЕРЕМЕННЫЕ

NORMAL = "normal"

Определяет цвет, используемый для некоторых слов, встречающихся в контекстах, отличных от перечисленных ниже.

BAD = "red"

Определяет цвет, используемый для некоторых слов, встречающихся в отрицательном контексте.

BRACKET = "blue"

Определяет цвет, используемый для скобок.

GOOD = "green"

Определяет цвет, используемый для некоторых слов, встречающихся в положительном контексте.

HILITE = "teal"

Определяет цвет, используемый для выделения слов.

INFORM = "darkgreen"

Определяет цвет, используемый для значений.

MERGE_LIST_PROGRESS = "yellow"

Определяет цвет, используемый для чисел, отображающих ход установки.

PKG_BLOCKER = "red"

Определяет цвет, используемый для пакетов, создающих неразрешенный конфликт.

PKG_BLOCKER_SATISFIED = "darkblue"

Определяет цвет, используемый для пакетов, создавших конфликт, который затем был разрешен.

PKG_MERGE = "darkgreen"

Определяет цвет, используемый для пакетов, которые будут установлены.

PKG_MERGE_SYSTEM = "darkgreen"

Определяет цвет, используемый для system-пакетов, которые будут установлены.

PKG_MERGE_WORLD = "green"

Определяет цвет, используемый для world-пакетов, которые будут установлены.

PKG_BINARY_MERGE = "purple"

Определяет цвет, используемый для пакетов, которые будут установлены в бинарной версии.

PKG_BINARY_MERGE_SYSTEM = "purple"

Определяет цвет, используемый для system-пакетов, которые будут установлены в бинарной версии.

PKG_BINARY_MERGE_WORLD = "fuchsia"

Определяет цвет, используемый для world-пакетов, которые будут установлены в бинарной версии.

PKG_NOMERGE = "darkblue"

Определяет цвет, используемый для имен пакетов, которые не будут установлены.

PKG_NOMERGE_SYSTEM = "darkblue"

Определяет цвет, используемый для имен system-пакетов, которые не будут установлены.

PKG_NOMERGE_WORLD = "blue"

Определяет цвет, используемый для имен world-пакетов, которые не будут установлены.

PKG_UNINSTALL = "red"

Определяет цвет, используемый для имен пакетов, которые должны быть удалены для разрешения конфликтов.

PROMPT_CHOICE_DEFAULT = "green"

Определяет цвет, используемый для предлагаемого на выбор значения по умолчанию.

PROMPT_CHOICE_OTHER = "red"

Определяет цвет, используемый для предлагаемого на выбор значения не по умолчанию.

SECURITY_WARN = "red"

Определяет цвет, используемый для предупреждений о безопасности.

UNMERGE_WARN = "red"

Определяет цвет, используемый для предупреждений об удалении пакета.

WARN = "yellow"

Определяет цвет, используемый для предупреждений.

ДОПУСТИМЫЕ АТРИБУТЫ

Цвет текста

black - черный

darkgray - темно-серый

darkred - темно-красный

red - красный

darkgreen - темно-зеленый

green - зеленый

brown - коричневый

yellow - желтый

darkyellow - светло-коричневый

darkblue - темно-синий

blue - синий

purple - фиолетовый

fuchsia - лиловый

teal - серо-зеленый

turquoise = **darkteal** - бирюзовый

lightgray - светло-серый

white - белый

Цвет фона

bg_black - черный фон

bg_darkred - темно-красный фон

bg_darkgreen - темно-зеленый фон

bg_brown = **bg_darkyellow** - коричневый фон

bg_darkblue - темно-синий фон

bg_purple - фиолетовый фон

bg_teal - серо-зеленый фон

bg_lightgray - светло-серый фон

Другие атрибуты

normal - обычный

no-attr - без атрибутов

reset - переопределить

bold - жирный

faint - бледный

standout - стандартный вывод

no-standout - не использовать стандартный вывод

underline - с подчеркиванием

no-underline - без подчеркивания

blink - мигающий

no-blink - без мигания

overline - с надчеркиванием

no-overline - без надчеркивания

reverse - негатив

no-reverse - не отражать цвет

invisible - невидимый

БАГТРЕКЕР

Об обнаруженных ошибках сообщайте на <http://bugs.gentoo.org/>

АВТОРЫ

- Arfrever Frehtes Taifersar Arahesis [\[Arfrever.FTA@gmail.com\]](mailto:Arfrever.FTA@gmail.com)(mailto:Arfrever.FTA@gmail.com)

ФАЙЛЫ

/etc/portage/color.map

Содержит переменные, используемые для пользовательской настройки цветного вывода.

/etc/make.conf

Содержит другие переменные.

СМ. ТАКЖЕ

console_codes(4), make.conf(5), portage(5), emerge(1), ebuild(1), ebuild(5)

Модуль Python */usr/lib/portage/pym/portage/output.py*.

ПЕРЕВОД

- Елена Гаврилова [\[e.vl.gavrilova@yandex.ru\]](mailto:e.vl.gavrilova@yandex.ru)(mailto:e.vl.gavrilova@yandex.ru)\

Март 2010

DISPATCH-CONF

НАЗВАНИЕ

dispatch-conf - безопасное обновление конфигурационных файлов после установки новых пакетов

СИНТАКСИС

dispatch-conf

ОПИСАНИЕ

Утилиту *dispatch-conf* следует запускать после установки новых пакетов для проверки конфигурационных файлов на обновления. Если новый конфигурационный файл попытается затереть текущий, *dispatch-conf* предложит пользователю самому решить, каким образом разрешить эту ситуацию. Среди достоинств *dispatch-conf* - легкость отката изменений (изменения конфигурационных файлов сохраняются с помощью либо патчей, либо RCS) и возможность автоматического обновления тех файлов, которые пользователь не изменял, и тех, которые отличаются от текущей версии только CVS-мусором или пробелом.

dispatch-conf проверит на обновления все каталоги, указанные в переменной **_CONFIG_PROTECT_**, и автоматически обновит все файлы, фигурирующие в **_CONFIG_PROTECT_MASK_**. Подробнее см. в **make.conf(5)**.

ОПЦИИ

Нет.

СИНТАКСИС

dispatch-conf следует запускать от администратора, поскольку владельцем файлов, с которыми работает утилита, как правило, является именно пользователь root. Перед первым запуском *dispatch-conf* необходимо отредактировать настройки в файле **/etc/dispatch-conf.conf** и создать каталог архивов, указанный в **/etc/dispatch-conf.conf**. Все изменения конфигурационных файлов сохраняются в каталоге архивов - либо как патчи, либо с помощью RCS, благодаря чему довольно просто вернуться к предыдущей версии.

Всякий раз, когда *dispatch-conf* обнаруживает конфигурационный файл, который был обновлен, пользователю дается возможность выбрать один из следующих вариантов, чтобы решить, что делать с предлагаемым обновлением:

u

Обновить (заменить) текущий конфигурационный файл новым и продолжить.

z

Затереть (удалить) новый конфигурационный файл и продолжить.

n

Пропустить и перейти к следующему конфигурационному файлу, не удаляя ни исходную версию, ни файлы, защищенные **_CONFIG_PROTECT_**.

e

Редактировать новый конфигурационный файл в редакторе текста, определенном переменной **\$EDITOR**.

m

В интерактивном режиме произвести слияние текущего и нового конфигурационных файлов.

I

Просмотреть различия между текущим и новым конфигурационными файлами.

t

Переключаться между текущим и новым конфигурационными файлами (в конечном итоге потребуется установить конечную версию, нажав u).

h

Вывести справку.

q

Выйти из *dispatch-conf*.

ФАЙЛ

ВНИМАНИЕ: Если **/etc/dispatch-conf.conf** сконфигурирован для использования **rcs(1)**, права на чтение и исполнение архивированных файлов могут быть унаследованы от первой проверки рабочего файла, как описано в man-руководстве **ci(1)**. Это означает, что даже если права доступа к рабочему файлу изменились, прежние права, действовавшие при первой проверке, могут быть возвращены. Согласно руководству **ci(1)**, пользователи могут управлять доступом к RCS-файлам, изменив права на доступ к каталогу, в котором они лежат.

БАГТРЕКЕР

Об обнаруженных ошибках сообщайте на <http://bugs.gentoo.org/>

АВТОРЫ

- Jeremy Wohl
- Karl Trygve Kalleberg [\(mailto:karltk@gentoo.org\)](mailto:karltk@gentoo.org)
- Mike Frysinger [\(mailto:vapier@gentoo.org\)](mailto:vapier@gentoo.org)
- Grant Goodyear [\(mailto:g2boojum@gentoo.org\)](mailto:g2boojum@gentoo.org)

ФАЙЛЫ

/etc/dispatch-conf.conf

Здесь хранятся настройки конфигурации для *dispatch-conf*.

СМ. ТАКЖЕ

[make.conf\(5\)](#), [ci\(1\)](#), [etc-update\(1\)](#), [rcs\(1\)](#)

ПЕРЕВОД

- Елена Гаврилова [\(mailto:e.vl.gavrilova@yandex.ru\)](mailto:e.vl.gavrilova@yandex.ru)

Январь 2011

EBUILD

НАЗВАНИЕ

ebuild - низкоуровневый интерфейс системы портежей

СИНТАКСИС

ebuild *файл команда* [*команда*] ...

ОПИСАНИЕ

Программа ebuild представляет собой прямой интерфейс системы Portage. Она обеспечивает возможность непосредственного взаимодействия с ебилдом при помощи специальных подкоманд или групп команд, выполняемых в контексте данного ебилда, и функций. Утилита ebuild принимает в качестве аргументов ebuild-сценарий и одну или более команд, подвергает сценарий синтаксическому анализу и выполняет указанные команды. Имеются отдельные команды для загрузки исходных файлов, их распаковки, компиляции, установки объектных файлов во временный каталог *image*, установки образа в локальную файловую систему, создания архива пакета из образа и т.д.

ФАЙЛ

Должен быть рабочим ebuild-сценарием. Подробнее см. в руководстве по [ebuild\(5\)](#).

КОМАНДЫ

По умолчанию portage выполняет все функции по порядку вплоть до указанной. Например, если вы дадите команду **compile**, то вызовете тем самым и предшествующие ей функции (такие как **setup** и **unpack**). Если вы хотите, чтобы запускалась только одна команда, вам следует добавить опцию *poauto* в значение переменной окружения **FEATURES**. Подробнее см. в справке по [make.conf\(5\)](#).

help

Выводит справочную информацию о программе в сжатом изложении, а также целый ряд сведений о пакете.

setup

Запускает все действия по настройке данного пакета, в том числе специфические системные тесты.

clean

Очищает временный каталог сборки, созданный Portage специально для этого ебилда. Временный каталог `build` обычно содержит извлеченный из архива исходный код, а также, возможно, так называемый установочный образ (все файлы, которые будут установлены в локальную файловую систему или сохранены в пакете). Расположение каталога сборки определяется значением переменной `PORTRAGE_TMPDIR`. Чтобы узнать ее текущее значение, выполните `emerge --info`. О том, как переопределить эту переменную, см. в справке по [make.conf\(5\)](#).

Примечание: Portage удаляет практически все данные, оставшиеся после успешной установки пакета, за исключением тех случаев, когда в переменной `FEATURES` явно указано `noclean`. Если вы добавите `noclean` в значение `FEATURES`, очень скоро большой объем дискового пространства будет занят ненужными файлами. Не рекомендуется пользоваться этим режимом постоянно, а лишь в том случае, если исходники пакетов потребуются вам после установки. Впрочем, возможно и ручное удаление этих файлов: для этого следует выполнить `rm -rf /var/tmp/portage`.

fetch

Проверяет, все ли источники данных, фигурирующие в `SRC_URI`, доступны в каталоге `DISTDIR` (подробнее см. в [make.conf\(5\)](#)) и имеют верную контрольную сумму. Если исходные коды недоступны, будет предпринята попытка загрузить их с серверов, адреса которых указаны в `SRC_URI`. Если для того или иного файла имеется несколько адресов загрузки, Portage проверит каждый из них и выберет тот сервер, который ближе. (Точность этого выбора на данный момент не гарантируется.) В первую очередь всегда обрабатываются зеркала Gentoo Linux, содержащиеся в переменной `GENTOO_MIRRORS`. Если по какой-либо причине md5-дайджест текущих или только что загруженных исходных кодов не совпадает с дайджестом в `files/digest-[пакет]-[версия-ревизия]`, выводится предупреждение, и ебилд завершает работу с кодом ошибки 1.

digest

В настоящее время - эквивалент команды `manifest`.

manifest

Обновляет Manifest-файл пакета. В результате создаются контрольные суммы для всех файлов, обнаруженных в одном каталоге с обрабатываемым ебилдом, а также содержимое вложенных каталогов подкаталога `files`. При этом контрольные суммы генерируются и для всех файлов, перечисленных в `SRC_URI` для каждого ебилда. Подробнее о поведении данной команды, см. в разделе о смысле значения `assume-digests` переменной `FEATURES` справочного руководства по [make.conf\(5\)](#). Если вы не хотите, чтобы дайджесты принимались неявно, см. опцию `--force`.

unpack

Извлекает исходные коды в подкаталог каталога `build` (`BUILD_PREFIX`), вызывая функцию `_src_unpack()` внутри файла ебилда. Если параметры функции `src_unpack()` не были указаны, для распаковки всех файлов, перечисленных в `SRC_URI`, будет использована стандартная `src_unpack()`. Как правило, исходники распаковываются в каталог `${BUILD_PREFIX}/[пакет]-[версия-ревизия]/work`. Обращаться к нему можно с помощью переменной `${WORKDIR}`.

Создавая ебилд самостоятельно, убедитесь, что переменная `S` (каталог исходных файлов), определенная в начале ebuild-сценария, указывает на каталог, в котором действительно содержатся распакованные исходные коды. По умолчанию он определяется как `${WORKDIR}/ ${P}`, поэтому, как правило, ничего не требуется исправлять. Функция `src_unpack()` также отвечает за наложение патчей перед компиляцией пакетов.

prepare

Подготавливает извлеченные из архива исходные коды, вызывая функцию `_src_prepare()`, параметры запуска которой определены в ebuild-файле. При запуске `src_prepare()` текущим рабочим каталогом становится `${S}`. Данная функция поддерживается начиная с `EAPI 2`.

configure

Производит конфигурирование распакованных исходных кодов, вызывая функцию `_src_configure()`, параметры запуска которой определены в ebuild-файле. При запуске `src_configure()` текущим рабочим каталогом становится `$_S`. Данная функция поддерживается начиная с **EAPI 2**.

compile

Компилирует распакованные исходные коды, вызывая функцию `_src_compile()`, параметры запуска которой определены в ebuild-файле. При запуске `src_compile()` текущим рабочим каталогом становится `$_S`. По завершении работы `src_compile()` исходные коды должны быть полностью скомпилированы.

test

Выполняет специальные тесты для отдельных пакетов, проверяя сборку.

preinst

Выполняет специальные действия для отдельных пакетов, которые требуется произвести до установки пакета в текущую файловую систему.

install

Устанавливает пакет во временный каталог `install`, вызывая функцию `_src_install()`. По завершении каталог установки в `(${BUILD_PREFIX}/[пакет]-[версия-ревизия]/image)` будет содержать все файлы, которые должны быть либо установлены в текущую файловую систему, либо включены в бинарный пакет.

postinst

Выполняет специальные действия для отдельных пакетов, которые требуется произвести после установки пакета в текущую файловую систему. Как правило, при этом выводятся полезные сообщения.

qmerge

Эта функция устанавливает все файлы в каталог `install` в текущей файловой системе. Это производится следующим образом: сначала запускается функция `_pkg_preinst()` (если указана). Затем все файлы устанавливаются в файловую систему, а их дайджесты контрольной суммы записываются в `/var/db/pkg/${CATEGORY}/${PN}-${PVR}/CONTENTS_`. Наконец, по завершении установки всех файлов выполняется функция `_pkg_postinst()` (если указана).

merge

Обычно для установки ебилда, необходимо последовательно выполнить следующие действия: `fetch`, `unpack`, `compile`, `install` и `qmerge`. Если вам нужно только установить ебилд, вы можете использовать данную команду: она сама выполнит все перечисленные операции и остановится в процессе выполнения только в том случае, если какая-либо функция отрабатывает с ошибкой.

unmerge

Эта команда сначала вызывает функцию `_pkg_prerm()` (если та указана). Затем она удаляет все файлы из текущих файловых систем, файл содержимого пакета для которых имеет неверные контрольную сумму и время изменения. Все пустые каталоги удаляются вместе с вложенными. Наконец, команда запускает функцию `_pkg_postrm()` (если та указана). Можно сначала установить новую версию пакета, а затем удалить прежнюю - собственно, именно в этом заключается рекомендуемый метод обновления.

prerm

Запускает для определенного пакета действия, которые необходимо выполнить до удаления пакета из файловой системы. См. также `unmerge`.

postrm

Запускает для определенного пакета действия, которые необходимо выполнить после удаления пакета из файловой системы. См. также `unmerge`.

config

Запускает для определенного пакета действия, которые необходимо выполнить до начала установки. Как правило, это настройка конфигурационных файлов или другие настроечные действия, назначаемые пользователем.

package

Эта команда очень напоминает *merge*, за исключением того, что после загрузки, распаковки, компиляции и установки создается .tbz2-архив бинарного пакета, который затем сохраняется в каталоге **PKGDIR** (см. [make.conf\(5\)](#)).

rpm

Собирает RPM-пакет RedHat из файлов во временном каталоге *install*. На данный момент сведения о зависимостях ебилда не включаются в RPM.

ОПЦИИ

--debug

Запустить bash с опцией -x option, в результате чего стандартный вывод будет включать подробную отладочную информацию.

--color < y | n >

Включить или отключить цветное отображение. Эта опция переопределяет значение переменной *NOCOLOR* (см. [make.conf\(5\)](#)) и может быть использована для принудительного назначения цвета в том случае, если стандартный вывод - не терминал (по умолчанию цвет включен только в том случае, если стандартный вывод - терминал).

--force

При использовании в связке с командой *digest* или *manifest* данная опция принудительно генерирует новые дайджесты для всех файлов исходного кода, относящихся к данному ебилду. Если в каталоге \${*DISTDIR*} требуемых исходников нет, они будут автоматически загружены.

--ignore-default-opts

Не использовать переменную окружения _EBUILD_DEFAULT_OPTS_.

--skip-manifest

Пропустить проверку Manifest-файлов.

БАГТРЕКЕР

Об обнаруженных ошибках сообщайте на <http://bugs.gentoo.org/>

АВТОРЫ

- Achim Gottinger [\[achim@gentoo.org\]\(mailto:achim@gentoo.org\)\](mailto:achim@gentoo.org)
- Daniel Robbins [\[drobbins@gentoo.org\]\(mailto:drobbins@gentoo.org\)\](mailto:drobbins@gentoo.org)
- Nicholas Jones [\[carpaski@gentoo.org\]\(mailto:carpaski@gentoo.org\)\](mailto:carpaski@gentoo.org)
- Mike Frysinger [\[vapier@gentoo.org\]\(mailto:vapier@gentoo.org\)\](mailto:vapier@gentoo.org)

ФАЙЛЫ

/etc/make.conf

Содержит переменные сборки, имеющие приоритет перед значениями, указанными в файле *make.globals*.

/etc/portage/color.map

Содержит переменные, позволяющие назначать пользовательские настройки цветного вывода.

СМ. ТАКЖЕ

[emerge](#) (1), [ebuild](#) (5), [make.conf](#) (5), [color.map](#) (5)

Сценарий `/usr/sbin/ebuild.sh`.

Вспомогательные приложения в каталоге `/usr/lib/portage/bin`.

ПЕРЕВОД

- Елена Гаврилова [\[e.vl.gavrilova@yandex.ru\]\(mailto:e.vl.gavrilova@yandex.ru\)](mailto:e.vl.gavrilova@yandex.ru)

Февраль 2011

EBUILD (5)

НАЗВАНИЕ

`ebuild` - внутренний формат, переменные и функции в сценарии `ebuild`

ОПИСАНИЕ

Программа **`ebuild`** (1) может принять в качестве аргумента только один сценарий `ebuild`. Сценарий `ebuild` содержит переменные и команды, которые определяют, каким образом следует загружать исходный код того или иного программного пакета, распаковывать его, накладывать патчи, компилировать, устанавливать и включать в дерево. Кроме того, в сценарий также входят команды, которые должны выполняться до и после установки/удаления пакетов. Все сценарии `ebuild` написаны на `bash`.

АТОМЫ ЗАВИСИМОСТИ

Атомы зависимости представляют собой не что иное как зависимости, к которым будет обращаться `portage`, выявляя отношения между пакетами. Обратите внимание, что если атом еще не в дереве пакетов, значение переменной совпадает с последней доступной версией.

Базовые атомы

Базовый атом - это полное имя пакета: категория/пакет.

Вот примеры базовых атомов:

```
sys-apps/sed  
sys-libs/zlib  
net-misc/dhcp
```

Версии атомов

Иногда необходимо уточнение, что подойдут лишь определенные версии атомов. Обратите внимание, что в этом случае их нужно указывать с префиксом (см. ниже). Иными словами, номер версии просто добавляется как суффикс к базовому атому.

Примеры:

```
sys-apps/sed-4.0.5  
sys-libs/zlib-1.1.4-r1  
net-misc/dhcp-3.0_p2
```

Номера версий обычно состоят из двух или трех чисел, разделенных точками, например, 1.2 или 4.5.2. После этой строки может идти буква - например, возможны такие номера версий как 1.2a или 4.5.2z. Имейте в виду, что эта буква *не обозначает* альфа- или бета-версию. Для этого используется специальный необязательный суффикс: `_alpha`, `_beta`, `_pre` (предрелиз), `_rc` (кандидат в релизы) или `_p` (патч). Таким образом, третий предрелиз пакета будет обозначен, например, как 1.2_pre3. Суффиксы можно добавлять один за другим в неограниченном количестве.

Префиксные операторы атома [> >= =

Иногда необходимо задать зависимость от группы версий, не указывая явно каждую из них. Для этого удобно использовать стандартные булевые операторы.

Примеры:

```
>media-libs/libgd-1.6  
>=media-libs/libgd-1.6  
=media-libs/libgd-1.6  
<=media-libs/libgd-1.6  
<media-libs/libgd-1.6
```

Расширенные префиксы [!~]и суффиксы атома [*]

Для большей точности возможно указывать блокирующие пакеты и соответствующие им диапазоны версий. Обратите внимание, что эти дополнительные префиксы/суффиксы можно сочетать любым способом с классами атомов, о которых шла речь выше.

~

означает любую ревизию указанной базовой версии. Так, в приведенном выше примере в выборке окажутся версии '1.0.2a', '1.0.2a-r1', '1.0.2a-r2' и т.д.

Пример:

```
-net-libs/libnet-1.0.2a
```

!

предотвращает одновременную установку пакетов, блокирующих друг друга.

Пример:

```
!app-text/dos2unix
```

!!

предотвращает одновременную установку пакетов и при этом явно запрещает их одновременную временную установку в серии обновлений. Данный синтаксис поддерживается в **EAPI 2** и выше.

Пример:

```
!!<sys-apps/portage-2.1.4_rc1
```

*

означает любую версию пакета при условии совпадения указанного префикса строки. Таким образом, если в качестве версии указано '2*', будут учитываться версии '2.1', '2.2', '2.2.1' и т.п., но не '1.0', '3.0', '4.1' и т.д. Не забудьте: в силу того, что вы используете подстановочный знак, шаблону '2*' будет соответствовать и '20'. Часть версии перед '*' должна быть корректным номером версии, без '*'.. Например, '2' - корректный номер версии, а '2.' - нет. Иными словами, вы можете использовать обозначение '2*', но не '2.*'.

Примеры:

```
=dev-libs/glib-2*  
!=net-fs/samba-2*
```

Слоты атомов

Начиная с **EAPI 1** может быть задан **слот** любого атома. Для этого необходимо указать имя **слота**, через двоеточие.

Например:

```
x11-libs/qt:3  
~x11-libs/qt-3.3.8:3  
>=x11-libs/qt-3.3.8:3  
=x11-libs/qt-3.3*:3
```

Подслоты

Начиная с **EAPI 5** зависимость слота может содержать optionalный подслот, который указывается через слэш /.

Примеры:

```
dev-libs/icu:0/0
dev-libs/icu:0/49
dev-lang/perl:0/5.12
dev-libs/glib:2/2.30
```

Операторы атомов слотов

Начиная с **EAPI 5**, оператором слота выступает двоеточие, за которым следует один из следующих операторов:

*

Означает, что разрешается любое значение слота. Кроме того, для сборочных зависимостей обозначает, что пакет, совпадающий с запросом, может быть удалён и заменён другим совпадающим в другом слоте.

Примеры:

```
dev-libs/icu:*
dev-lang/perl:*
dev-libs/glib:*
```

=

Означает, что допускается любое значение слота. Кроме того, для сборочных зависимостей означает, что слот и подслот пакета, совпадающего с запросом, обязательно должны совпадать со слотом и подслотом последней присутствующей в системе на момент установки версии.

Примеры:

```
dev-libs/icu:=
dev-lang/perl:=
dev-libs/glib:=
```

slot=

Означает то же, что и оператор равенства, за исключением того, что для слота допускается лишь одно определённое значение.

Примеры:

```
dev-libs/icu:0=
dev-lang/perl:0=
dev-libs/glib:2=
```

Чтобы использовать оператор равенства слота, пакетный менеджер должен сохранить пару слот/подслот новейшей установленной версии пакета. Описанный синтаксис обрабатывается только пакетным менеджером и не предназначен для билдов. Для этого пакетный менеджер при сохранении зависимостей пакета может вставлять соответствующий слот/подслот между двоеточием и знаком равенства. Обозначение подслота в этом случае опускать нельзя (когда это делает переменная SLOT, подразумевается, что пакет имеет подслот, совпадающий со слотом).

Примеры:

```
dev-libs/icu:0/0=
dev-libs/icu:0/49=
dev-lang/perl:0/5.12=
```

USE-флаги атомов

Начиная с **EAPI 2** для любого атома могут быть заданы USE -флаги. Если при этом вы также указываете **слоты**, то зависимости USE -флагов будут выводится справа от зависимостей **слотов**.

Безусловные USE-зависимости Пример Расшифровка

`foo[bar]` foo требует, чтобы был включен bar

`foo[bar,baz]` foo требует, чтобы были включены и bar, и baz

`foo[-bar,baz]` foo требует, чтобы bar был отключен, а baz включен

Условные USE-зависимости Краткая запись Развернутая запись

`foo[bar?]` bar? (`foo[bar]`) !bar? (`foo`)

`foo[!bar?]` bar? (`foo`) !bar? (`foo[-bar]`)

`foo[bar=]` bar? (`foo[bar]`) !bar? (`foo[-bar]`)

`foo[!bar=]` bar? (`foo[-bar]`) !bar? (`foo[bar]`)

USE-флаги атомов по умолчанию

Начиная с **EAPI 4** USE- зависимости умеют устанавливать значения по умолчанию для флагов, которые могут быть не прописаны в **IUSE** для данного пакета. Для этого непосредственно вслед за флагом поставьте соответственно (+) или (-): (+) указывает системе вести себя так, словно отсутствующий флаг присутствует и включен, (-) - так, словно отсутствующий флаг присутствует, но отключен.

`media-video/ffmpeg[threads(+)]`

`media-video/ffmpeg[threads(-)]`

Динамические зависимости

Зачастую программы могут иметь различные зависимости при различных USE-флагах. Система портежей предлагает несколько опций для учета этого. Заметьте, что каждая синтаксическая конструкция из числа описанных ниже каждая конструкция рассматривается как один атом в соответствующем контексте. Таким образом, каждый атом за счет задания условий может включать несколько атомов, и эта вложенность может быть любой глубины.

переменная_use? (атом)

Так, чтобы включить библиотеку jpeg, при указании пользователем jpeg в переменной USE, используйте следующую конструкцию:

`jpeg? (media-libs/jpeg)`

!переменная_use? (атом)

Если пакет необходимо включать только при отсутствии определенной опции в пользовательском значении переменной USE, используйте конструкцию, подобную:

`!nophysfs? (dev-games/physfs)`

Это удобно в том случае, если вы хотите добавить необязательную поддержку той или иной возможности и устанавливать ее по умолчанию.

переменная_use? (атом, если истинно) !переменная_use? (атом, если ложно)

Для обработки аналогично тернарному оператору языка C используйте два утверждения, одно прямое, другое обратное. Так, если пакет использует GTK2 или GTK1, но не оба вместе, это можно выразить следующим образом:

`gtk2? (=x11-libs/gtk+-2*) !gtk2? (=x11-libs/gtk+-1*)`

В результате по умолчанию будет использоваться более новая библиотека, GTK2.

|| (Atom Atom ...)

Если пакет может работать с несколькими различными пакетами, но категория `virtual` не подходит, может быть использован следующий синтаксис:

```
|| (
    app-games/unreal-tournament
    app-games/unreal-tournament-goty
)
```

В приведенном примере мы видим, что пакет `unreal-tournament` существует в обычной версии и в версии `goty`. Поскольку обе они предоставляют один и тот же базовый набор файлов, другой пакет может использовать любой из них. Добавление категории `virtual` не подходит, поскольку она не охватывает соответствующие файлы.

Другой уместный случай использования данной конструкции - это когда пакет может быть собран с различными видео-интерфейсами, но не может использовать их одновременно.

Например:

```
|| (
    sdl? ( media-libs/libsdl )
    svga? ( media-libs/svgalib )
    opengl? ( virtual/opengl )
    ggi? ( media-libs/libggi )
    virtual/x11
)
```

В данном случае будет выбран только один из пакетов, а приоритет определяется порядком их перечисления. Поэтому наиболее вероятным кандидатом является библиотека `sdl`, далее идут `svga`, `opengl`, `ggi` - и, наконец, по умолчанию, если пользователь не задал явно одну из предлагаемых библиотек, будет использоваться `X`.

Имейте в виду, что, если один из перечисленных пакетов уже установлен, менеджер пакетов будет руководствоваться этим для удовлетворения зависимости.

Перекрёстная компиляция

Portage поддерживает перекрёстную компиляцию в каталоге, определяемом переменной **ROOT**.

Хост

В данном случае под **хостом** понимается платформа, на которой разворачивается процесс сборки - в системе сборки GNU это значение переменной `CBUILD`. Пакеты хоста хранятся в корневом каталоге системы ("").

Если **ROOT** - "", сюда будут устанавливаться все типы зависимостей. Если это не так, для EAPI, поддерживающих переменную **HDEPEND** (экспериментальный **EAPI 5-hdepend**), в каталог "/" будут устанавливаться только пакеты **HDEPEND**. Для EAPI без поддержки **HDEPEND** это поведение управляет опцией `t --root-deps` команды `emerge(1)`; по умолчанию в `_хост_` будут устанавливаться только пакеты **DEPEND**.

Цель

Под **целью** подразумевается платформа, на которой пакет в дальнейшем будет работать - в системе сборки GNU это значение переменной `CHOST`. Каталог, где располагается данная система, определяется переменной **ROOT**. Если этот каталог - не "", то есть если **хост** и **цель** не совпадают, то путь, который хранится в переменной, указывает на каталог, в котором находится **цель**.

Для EAPI с поддержкой **HDEPEND** (экспериментальный EAPI 5-hdepend), переменные **DEPEND**, **RDEPEND** и **PDEPEND** содержат списки зависимостей *цели* - иными словами, перечни пакетов, которые должны быть установлены в каталог **ROOT**. Для EAPI, которые не поддерживают **HDEPEND**, за установку в этот каталог отвечает опция `--root-deps` пакетного менеджера **emerge(1)**. Если данная опция не включена, **emerge** по умолчанию будет устанавливать сборочные зависимости (то есть пакеты **RDEPEND** и **PDEPEND**) только в каталог **ROOT**.

Подробнее о переменных **DEPEND**, **RDEPEND** и **HDEPEND** см. в разделе **ПЕРЕМЕННЫЕ**.

USE-флаг targetroot

Для EAPI, которые поддерживают USE-флаг "targetroot", пакетный менеджер автоматически включает данный флаг, если *host* и *target* не совпадают - иными словами, если значение **ROOT** - не "/". Это необходимо в том случае, если пакету в процессе сборки требуется исполняемая копия себя самого. Распространённым примером такой ситуации может служить пакет `dev-lang/python`, который при компиляции требует запуска интерпретатора Python.

ПЕРЕМЕННЫЕ

Замечания по синтаксису

- В сценариях ebuild можно использовать все переменные, определенные в [make.conf\(5\)](#) (такие как **PORTAGE** и **PORTDIR**)
- Присваивая значения переменным в сценарии, **нельзя оставлять пробел** между именем переменной и знаком равенства.
- Значения переменных могут содержать только символы из таблицы [ascii\(7\)](#). Это требование **GLEP 31**.

P

Эта переменная содержит имя пакета без указания ревизии сценария. Ее НЕЛЬЗЯ изменять.

```
xfree-4.2.1-r2.ebuild --> $P=='xfree-4.2.1'
```

PN

Содержит имя сценария без номера версии.

```
xfree-4.2.1-r2.ebuild --> $PN=='xfree'
```

PV

Содержит номер версии без указания ревизии.

```
xfree-4.2.1-r2.ebuild --> $PV=='4.2.1'
```

PR

Содержит номер ревизии или, если его не существует, 'r0'.

```
xfree-4.2.1-r2.ebuild --> $PR=='r2'
```

PVR

Содержит номер версии с указанием ревизии.

```
xfree-4.2.1-r2.ebuild --> $PVR=='4.2.1-r2'
```

PF

Содержит полное имя пакета **PN-PVR**

```
xfree-4.2.1-r2.ebuild --> $PF=='xfree-4.2.1-r2'
```

КАТЕГОРИЯ

Содержит имя категории пакета.

A

Содержит все исходные файлы, необходимые пакету. Эта переменная не должна определяться: она генерируется автоматически на основе переменной **SRC_URI**.

WORKDIR = `_${PORTAGE_TMPDIR}/portage/${CATEGORY}/${PF}/work_`

Содержит путь доступа к корневому каталогу сборки пакета. Не изменяйте эту переменную.

FILESDIR = `_${PORTDIR}/${CATEGORY}/${PN}/files`

Содержит путь доступа к подкаталогу 'files' в каталоге пакета в дереве портежей. Не изменяйте эту переменную.

EBUILD_PHASE

Contains the abbreviated name of the phase function that is currently executing, such as "setup", "unpack", "compile", or "preinst".

EBUILD_PHASE_FUNC

Начиная с **EAPI 5** содержит полное имя выполняемой в данный момент фазовой функции, например, "pkg_setup", "src_unpack", "src_compile", "pkg_preinst".

EPREFIX

Начиная с **EAPI 3** для этой переменной определено смещение, для которого был конфигурирован данный portage в ходе установки. Смещение иногда необходимо в ебилдах и екласах; в этом случае оно определяется как `_${EPREFIX}`. EPREFIX не содержит слэша, поэтому отсутствию смещения соответствует пустая строка. Не изменяйте эту переменную.

S = `_${WORKDIR}/${P}`

Содержит путь доступа к временному каталогу сборки. Эта переменная используется функциями `_src_compile_` и `_src_install_`. Обе они выполняются в каталоге *S*. Данная переменная может быть изменена, если необходимо указать другой каталог для распаковки архива пакета.

T = `_${PORTAGE_TMPDIR}/portage/${CATEGORY}/${PF}/temp_`

Содержит путь доступа к временному каталогу. Может использоваться в любых целях.

D = `_${PORTAGE_TMPDIR}/portage/${CATEGORY}/${PF}/image/_`

Содержит путь доступа к временному каталогу для установки. Каждая операция записи, выполняемая не через вспомогательные утилиты и функции (они описаны ниже), должна выполняться в каталоге `_${D}`. Начиная с **EAPI 3** здесь, как правило, приходится учитывать префикс смещения, для которого отведена переменная `_${ED}` (см.ниже). Не изменяйте эту переменную.

ED = `_${PORTAGE_TMPDIR}/portage/${CATEGORY}/${PF}/image/${EPREFIX}/_`

Начиная с **EAPI 3** для удобства содержит путь "`_${D%}/${EPREFIX}/`". Для версий **EAPI** ниже **EAPI 3**, не поддерживающих `_${ED}`, в тех случаях, когда в новых версиях используется `_${ED}`, вспомогательные функции обращаются к `_${D}`. Не изменяйте эту переменную.

MERGE_TYPE

Начиная с **EAPI 4** переменная **MERGE_TYPE** может использоваться для запроса текущего типа установки. Для нее запустимы следующие значения: Значение Расшифровка

binary прекомпилированный пакет назначен для установки

buildonly пакет будет скомпилирован из исходного кода, но не назначен для установки

source пакет будет скомпилирован из исходного кода и назначен для установки

PORTEAGE_LOG_FILE

Содержит путь доступа к журналу сборки. Если значение переменной **PORT_LOGDIR** не назначено, то **PORTEAGE_LOG_FILE** = `"_${T}/build.log"`.

REPLACED_BY_VERSION

Начиная с **EAPI 4** переменная `REPLACED_BY_VERSION` может использоваться функциями `pkg_prerm` и `pkg_postrm` для запроса версии пакета, которая заменит текущую. Переменная пуста, если нет нового пакета; если же такой пакет имеется, переменная будет содержать только номер версии.

REPLACING VERSIONS

Начиная с **EAPI 4** переменная `REPLACING_VERSIONS` может использоваться `pkg_pretend`, `pkg_setup`, `pkg_preinst` и `pkg_postinst` для запроса версии (версий) пакета, которую (которые) заменит данный пакет. Переменная пуста, если нет пакетов, которые подлежат замене; если есть хотя бы один такой пакет, переменная содержит через пробел, номера версии (версий) пакета, которые будет/-ут заменена/-ы. Как правило, эта переменная не содержит более одной версии, хотя, согласно PMS, может содержать несколько их.

ROOT = /

Содержит путь, по которому portage обращается к корневому каталогу текущей файловой системы. Когда пакеты вносят изменения в эту файловую систему, они должны использовать для этого дерево с префиксом `${ROOT}`. Часто при этом необходимо учитывать префикс смещения, за который отвечает переменная `${EROOT}` (см. ниже). Не изменяйте эту переменную.

EROOT = \${ROOT%/\$EPREFIX}/

Начиная с **EAPI 3** для удобства содержит `"${ROOT%/$EPREFIX}/"`. Не изменяйте эту переменную.

DESCRIPTION = A happy little package

Содержит краткое описание пакета.

EAPI = 0

Определяет версию API ебилда, которой соответствует данный пакет. Если значение не определено, то по умолчанию выставляется "0". Если portage не распознал значение EAPI, пакет будет замаскирован и какие-либо операции над ним будут невозможны; в этом случае сначала необходимо обновить portage. Для максимальной обратной совместимости пакет должен соответствовать как можно более ранней версии EAPI. Обратите внимание, что использование команд `ebuild(1)` и `repoman(1)` с пакетом требует версии portage, которая умеет работать с соответствующей данной пакету версией EAPI.

SRC_URI = [http://example.com/path/\\${P}.tar.gz](http://example.com/path/${P}.tar.gz)

Содержит перечень URI необходимых файлов исходного кода. Одному файлу при этом могут соответствовать несколько URI. Список обрабатывается в порядке следования элементов: если элемент не удается найти ни на одном из зеркал `_GENTOO_MIRRORS`, система переходит к следующему элементу. Начиная с **EAPI 2** имя файла вывода определенного URI может быть изменено по вашему желанию: для этого справа ставится оператор `"->"` и, вслед за ним, имя файла вывода, какое вы предпочитаете. Все элементы, в частности, оператор и имя файла вывода, разделяются пробелом.

HOMEPAGE = <http://example.com/>

Эта переменная должна содержать URI основных сайтов, с которых берутся исходные коды, и другую дополнительную информацию, в зависимости от пакета.

KEYWORDS = [-~][x86,ppc,sparc,mips,alpha,arm,hppa]

Эта переменная должна содержать корректный список архитектур, под которыми работает/не работает ебилд. По умолчанию, если неизвестно, требует ли ебилд определенной архитектуры, соответствующее значение KEYWORD просто не указывается. Если известно, что ебилд не работает под той или иной архитектурой, укажите ее с минусом, например, `-ppc`. Если ебилд посыпается для включения в дерево, он должен иметь ключ `~arch` для архитектур, на которых он ПРЕДПОЛОЖИТЕЛЬНО РАБОТАЕТ. (Чтобы размаскировать для тестирования пакеты, помеченные такими ключевыми словами, укажите `ACCEPT_KEYWORDS="~arch"` в командной строке или непосредственно в файле `make.conf(5)`) Полный список архитектур см. в `/usr/portage/profiles/arch.list`. Страйтесь соблюдать алфавитный порядок списка.

SLOT

Устанавливает слот для пакетов, для работы которых может понадобиться существование нескольких версий. Значение по умолчанию - **SLOT**="0". Если вы не вполне уверены в своих действиях, не пытайтесь экспериментировать с этой переменной. В любом случае, ее значение **НИКОГДА** не должно оставаться неопределенным.

Начиная с **EAPI 5** переменная **SLOT** может опционально содержать подслот; он указывается вслед за слотом, от которого отделяется слэшем /. Подслот должен представлять собой корректное имя слота. Подслоты используются в тех случаях, когда обновление пакета до новой версии с другим подслотом может потребовать пересборки пакетов-зависимостей. Если подслот в переменной **SLOT** не определён, подразумевается, что пакет имеет подслот, совпадающий с основным слотом. Подробнее об использовании подслотов см. в разделе **Операторы атомов слотов**.

LICENSE

Представляет собой перечень, через пробел, лицензий, под которые подпадает пакет. Этот список **должен** соответствовать содержимому /usr/portage/licenses/. Если той или иной лицензии нет в portage, следует добавить ее явно.

IUSE

Представляет собой практически исчерпывающий список USE-флагов, которые используются в вашем сценарии ebuild. Здесь не фигурируют только те USE-флаги, которые описывают архитектуру (см. **KEYWORDS**). Начиная с **EAPI 1** можно создавать настройки по умолчанию, включая или отключая USE -флаги путем выставления перед именами флагов соответственно + и -. Более подробно о порядке помещения в стек USE -флагов см. в разделе, посвященном переменной **USE_ORDER** в [make.conf\(5\)](#). С учетом значения по умолчанию **USE_ORDER**, отрицательные настройки IUSE по умолчанию действуют на USE только на уровне репозитория, поскольку параметры профиля и пользовательские настройки имеют над ними приоритет.

DEPEND

Данная переменная должна содержать перечень пакетов, необходимых программе для компиляции (они же сборочные зависимости). Как правило, это библиотеки и заголовки.

Начиная с экспериментального **EAPI 5-hdepend** список утилит следует хранить не здесь, а в переменной **HDEPEND**, поскольку пакеты **DEPEND** будут установлены в каталог цели, а значит, не смогут выполняться при перекрестной компиляции (подробнее см. в разделе **Перекрестная компиляция**).

Здесь вы можете использовать синтаксис, описанный в разделе **Зависимости**.

RDEPEND

Эта переменная должна содержать перечень всех пакетов, необходимых для работы данной программы (они же сборочные зависимости). Как правило, это библиотеки.

EAPI 3 и ниже, если она не определена, по умолчанию будет использовано значение переменной **DEPEND**. В **EAPI 4** и выше переменная **RDEPEND** никогда не задается явно.

Здесь вы можете использовать синтаксис, аналогичный описанному выше в разделе **Зависимости**.

HDEPEND

Эта переменная должна содержать перечень всех пакетов, которые должны быть доступны как исполняемые при сборке данной программы (они же сборочные зависимости). Как правило, это утилиты: интерпретаторы или (кросс-)компиляторы.

Эта переменная была введена в экспериментальном **EAPI 5-hdepend** и будет установлена в каталог хоста (подробнее см. в разделе **Перекрестная компиляция**).

Здесь вы можете использовать синтаксис, аналогичный описанному выше в разделе **Зависимости**.

PDEPEND

Эта переменная должна содержать перечень всех пакетов, которые следует установить после установки данного пакета (постсборочные зависимости), но фактически они могут быть установлены раньше.

ВНИМАНИЕ

Используйте эту возможность только в крайнем случае для прерывания циклических зависимостей!

Здесь вы можете использовать синтаксис, аналогичный описанному выше в разделе **Зависимости**.

REQUIRED_USE

Начиная с **EAPI 4** используется переменная **REQUIRED_USE**, позволяющая указывать допустимые и недопустимые комбинации **USE-** флагов. При необходимости можно использовать вложения. Действие Выражение

Если флаг1 включен, то флаг2 отключен флаг1? (!флаг2)

Если флаг1 включен, то флаг2 включен флаг1? (флаг2)

Если флаг1 отключен, то флаг2 включен !флаг1? (флаг2)

Если флаг1 отключен, то флаг2 отключен !флаг1? (!флаг2)

Следует включить любой из флагов (включающее ИЛИ) || (флаг1 флаг2 флаг3)

Следует включить только один определенный флаг (исключающее ИЛИ) ^ (флаг1 флаг2 флаг3)

(Начиная с **EAPI 5:**) Можно включить в крайнем случае один определённый флаг ?? (флаг1 флаг2 флаг3)

RESTRICT = [*strip,mirror,fetch,userpriv*]

Задает ограничения возможностей системы портежей, через пробел. Для задания динамических ограничений можно использовать условные конструкции, аналогичные описанным выше для переменной **DEPEND**.

binchecks

Отключает проверку бинарных файлов. Используйте это значение ТОЛЬКО в том случае, если такая проверка не имеет смысла (например, для оригинальных заголовков и исходных кодов ядра, которые не имеют бинарных версий). Если опрос бинарных файлов следует пропустить по другим причинам (например, если речь идет о проприетарных бинарных компонентах), см. подробнее в разделе **УПРАВЛЯЮЩИЕ QA-ПЕРЕМЕННЫЕ**.

bindist

Ограничение по собранным пакетам.

fetch

Аналогично *mirror*, но файлы не будут выбираться и через переменную **SRC_URI**.

installsources

Отключает *installsources* для определенных пакетов. Предназначено для пакетов, имеющих бинарные файлы, которые не совместимы с *debugedit*.

mirror

Файлы, указанные в переменной **SRC_URI**, не будут загружаться с зеркал **GENTOO_MIRRORS**

primaryuri

Файлы будут загружаться сначала с URI, указанных в переменной **SRC_URI**, и лишь потом с зеркал **GENTOO_MIRRORS**.

strip

Из результирующих двоичных модулей/библиотек не будет удаляться отладочная информация.

test

src_test не будет запускаться, даже если пользовательское значение **FEATURES=test**.

userpriv

Отключает *userpriv* для определенных пакетов.

PROPERTIES = [*interactive*]

Список параметров через пробел; поддерживается синтаксис условных конструкций.

interactive

На одной или нескольких фазах ебилда пользователю будет предложено ввести данные.

PROVIDE = "virtual/**ЦЕЛЬ**"

Эту переменную следует использовать, только если пакет имеет виртуальную цель. Например, blackdown-jdk и sun-jdk предоставляют virtual/jdk. Это позволяет пакетам задавать зависимости от virtual/jdk, а не явно от реализации blackdown или sun.

DOCS

Начиная с **EAPI 4** эта переменная содержит массив или разделенный пробелами список документационных файлов для функции src_install по умолчанию для установки с помощью dodoc. Если она не определена, будет использован разумный список по умолчанию. См. ниже справку по src_install.

УПРАВЛЯЮЩИЕ QA-ПЕРЕМЕННЫЕ

Замечания по синтаксису

Существует ряд QA-переменных, которые позволяют сценарию управлять некоторыми QA-тестами, выполняемыми portage. В ебилдах к ним следует прибегать как можно реже, поскольку в противном случае теряет смысл сам контроль качества. QA-переменные предназначены прежде всего для сценариев, которые устанавливают бинарные объекты из закрытых исходных кодов, которые не могут быть изменены.

Обратите внимание, что объекты, которые не соответствуют данным правилам, могут выдавать ошибку на некоторых архитектурах.

QA_PREBUILT

Содержит перечень путей доступа к прекомпилированным, бинарным файлам в каталоге образа. Указанные здесь пути добавляются к каждой из перечисленных ниже переменных QA_. Пути могут включать fnmatch-подобные шаблоны, которые встроеннымми средствами переводятся в регулярные выражения для переменных QA_: последние не поддерживают такие шаблоны, а регулярные выражения - поддерживают. Переводчик просто заменяет "*" на ".*".

QA_TEXTRELS

Может содержать перечень путей доступа к файлам в каталоге образа, которые содержат неустранимые text relocation. Пути могут включать шаблоны fnmatch.

Эта переменная предназначена для использования с бинарными объектами, установленными из закрытых исходных кодов, которые не могут быть изменены.

QA_EXECSTACK

Содержит список путей доступа к объектам в каталоге образа, которые для запуска требуют исполняемый стек. Пути могут включать шаблоны fnmatch.

Эта переменная предназначена для использования с объектами, которым действительно нужен исполняемый стек (а не с теми, которые лишь маркированы соответствующим образом).

QA_WX_LOAD

Эта переменная должна содержать список путей доступа к файлам в каталоге образа, которые содержат перезаписываемые и исполняемые сегменты - что случается редко. Пути могут содержать шаблоны fnmatch.

QA_FLAGS_IGNORED

Эта переменная должна содержать список путей доступа к файлам в каталоге образа, которые не содержат .GCC.command.line-секции или содержат .hash-секции. Пути могут содержать регулярные выражения с экранированными символами, заключенными в управляемые кавычки.

Переменная предназначена для использования с файлами бинарных пакетов, которые игнорируют переменные `CFLAGS`, `CXXFLAGS`, `FFLAGS`, `FCFLAGS`, `LDFLAGS`.

QA_MULTILIB_PATHS

Эта переменная содержит список путей доступа к файлам в каталоге образа, которые должны быть проигнорированы при проверках `multilib-strict`. Пути могут содержать регулярные выражения с экранированными символами, заключенными в управляемые кавычки.

QA_PRESTRIPPED

Эта переменная должна содержать список путей доступа к бинарным файлам в каталоге образа, не содержащим отладочной информации. Пути могут включать регулярные выражения с экранированными символами, заключенными в управляемые кавычки.

QA SONAME

Эта переменная должна содержать список путей доступа к файлам общедоступных библиотек в каталоге образа, не имеющих `SONAME`. Пути могут включать регулярные выражения с экранированными символами, заключенными в управляемые кавычки.

QA SONAME_NO_SYMLINK

Эта переменная должна содержать список путей доступа к файлам общедоступных библиотек в каталоге образа, имеющим `SONAME`, но символьской ссылки на `SONAME` в том же каталоге быть не должно. Пути могут включать регулярные выражения с экранированными символами, заключенными в управляемые кавычки.

QA_DT_NEEDED

Эта переменная должна содержать список путей доступа к файлам общедоступных библиотек в каталоге образа, не имеющим записей `NEEDED`. Пути могут включать регулярные выражения с экранированными символами, заключенными в управляемые кавычки.

QA_DESKTOP_FILE

Эта переменная должна содержать список путей доступа к desktop-файлам в каталоге образа, которые не должны учитываться. Пути могут включать регулярные выражения с экранированными символами, заключенными в управляемые кавычки.

ОБЪЯВЛЕНИЯ PORTAGE

inherit

`Inherit` обеспечивает поддержку в `portage` т.н. екласов функций, внешних по отношению к ебилдам и обладающих наследуемыми методами и данными. Они определяют функции и устанавливают типы данных как drop-in-замены, расширенные, упрощенные подпрограммы для самых обычных задач, с целью упорядочения процесса сборки. Вызов `inherit` не может зависеть от условий, которые могут меняться в пределах одного ебилда. Спецификация екласов содержит только их имена, без расширения `.eclass`. Обратите также внимание, что оператор `inherit` должен предшествовать объявлению других переменных, если только эти переменные не используются в глобальном контексте екласов.

ФАЗОВЫЕ ФУНКЦИИ

pkg_pretend

Эта функция может быть определена начиная с **EAPI 4**; она используется для проверки выполнения различных требований. Она должна вызываться как можно раньше, до того, как будет предпринята попытка удовлетворить зависимости. Если функция обнаружит проблему, она должна вызвать `eerror` и `die`. Окружение (переменные, функции, временные каталоги и т.д.), используемое для выполнения `pkg_pretend`, не сохраняется и, следовательно, не будет доступно на последующих фазах.

pkg_nofetch

Если в переменной **RESTRICT** указано *fetch*, эта функция будет запущена, если не удастся найти файлы в **SRC_URI**. Таким образом удобно сообщать пользователю, **как** получить данные файлы. Достаточно выдать сообщение и нормально завершить работу функции; не завершайте ее вызовом **die**.

pkg_setup

Эта функция может использоваться в том случае, если пакету требуется особая настройка или проверка, прежде чем будут выполняться другие действия.

Исходный рабочий каталог `${PORTAGE_TMPDIR}`.

src_unpack

Эта функция используется для распаковки всех исходных текстов, указанных в *A*, в каталог **WORKDIR**. Если эта функция в сценарии ebuild не определена, вызывается *unpack \${A}*. В этой функции производится наложение патчей и выполняются все остальные изменения, необходимые перед конфигурированием/компиляцией.

Исходный рабочий каталог `$WORKDIR`.

src_prepare

Все действия по подготовке исходного кода, в частности, наложение патчей, должны выполняться в этой функции. Она поддерживается в версиях EAPI начиная с **EAPI 2**.

Исходный рабочий каталог `$S`.

src_configure

Все необходимые действия по конфигурированию должны выполняться в этой функции. Она поддерживается в версиях EAPI начиная с **EAPI 2**.

Исходный рабочий каталог `$S`.

src_compile

В **EAPI 2** и ниже все необходимые шаги по конфигурированию и компиляции должны выполняться в этой функции; начиная с **EAPI 2** она отвечает лишь за компиляцию.

Исходный рабочий каталог `$S`.

src_test

Запускает все тест-кейсы, определенные для данного пакета. По умолчанию будет выполнено 'make check', а затем 'make test'.

Исходный рабочий каталог `$S`.

src_install

Должна содержать все необходимое для установки пакета во временном каталоге установки.

Исходный рабочий каталог `$S`.

Начиная с **EAPI 4**, если `_src_install_` не определена, используется следующее умолчание:

```
src_install() {
    if [[ -f Makefile || -f GNUmakefile || -f makefile ]] ; then
        emake DESTDIR="${D}" install
    fi

    if ! declare -p DOCS &>/dev/null ; then
        local d
        for d in README* ChangeLog AUTHORS NEWS TODO CHANGES \
                 THANKS BUGS FAQ CREDITS CHANGELOG ; do
            [[ -s "${d}" ]] && dodoc "${d}"
        done
    elif [[ $(declare -p DOCS) == "declare -a \"*\" " ]] ; then
        dodoc "${DOCS[@]}"
    else
        dodoc ${DOCS}
    fi
```

}

pkg_preinst **pkg_postinst**

В этих функциях должны выполняться все модификации текущей файловой системы, необходимые до и после включения пакета в дерево. Здесь также необходимо помещать комментарий для пользователя, выводимый в конце выполнения функции.

Исходный рабочий каталог \$PWD.

pkg_prerm **pkg_postrm**

Аналогично функциям pkg_*inst, но для исключения пакета из дерева.

Исходный рабочий каталог \$PWD.

pkg_config

Эта функция должна содержать необязательные основные шаги по конфигурированию.

Исходный рабочий каталог \$PWD.

ВСПОМОГАТЕЛЬНЫЕ ФУНКЦИИ

Фазы

default

Вызывает фазовые функции по умолчанию для текущей фазы выполнения. Эта функция поддерживается начиная с **EAPI 2**.

default_*

Начиная с **EAPI 2** функции pkg_nofetch и src_* с параметрами по умолчанию доступны через функцию с именем, начинающимся с default_ и заканчивающимся соответствующим именем фазовой функции. Например, вызвав функцию default_src_compile, вы фактически запускаете функцию src_compile с параметрами по умолчанию. Фазовые функции по умолчанию

default_pkg_nofetch
default_src_unpack
default_src_prepare
default_src_configure
default_src_compile
default_src_test

Разное

die [*причина*]

Вызывает прекращение работы текущего процесса emerge. Выводимая при этом информация будет включать причину.

Начиная с **EAPI 4** все вспомогательные функции автоматически вызывают **die** при любой ошибке. Вызовы вспомогательных функций могут иметь префиксом функцию **nonfatal**, дабы ошибки не рассматривались системой как фатальные.

nonfatal <вспомогательная функция>

Выполняет вспомогательную функцию, но в случае ошибки не вызывает **die**. Функция **nonfatal** доступна начиная с **EAPI 4**.

use <элемент USE>

Если элемент **USE** указан в переменной **USE**, функция без дополнительных сообщений возвращает 0 ("истинно"). Если же элемент **USE** в переменной **USE** не указан, функция без дополнительных сообщений возвращает 1 ("ложно"). Если вам нужен подробный вывод **use**, используйте вспомогательную функцию **usev**.

Пример:

```
if use gnome ; then
    guiconf="--enable-gui=gnome --with-x"
elif use gtk ; then
    guiconf="--enable-gui=gtk --with-x"
elif use X ; then
    guiconf="--enable-gui=athena --with-x"
else
    # gui-версия собираться не будет
    guiconf=""
fi
```

usex [вывод, если истинно] [вывод, если ложно] [суффикс, если истинно] [суффикс, если ложно]

Если USE-флаг установлен, то будет выполнено echo [true output][true suffix] (по умолчанию "yes"), иначе echo [false output][false suffix] (по умолчанию "no"). Функция **usex** доступна начиная с **EAPI 5**.

use_with <элемент USE> [имя configure] [опция configure]

Эта функция удобна для создания специфических опций, передаваемых сценарию configure. Если элемент *USE* указан в переменной **USE**, будет выведена строка **--with-[имя configure]=[опция configure]**. Если опция *configure* не указана, будет выведено только **--with-[имя configure]**. Если элемент *USE* не указан в переменной **USE**, будет выведена строка **--without-[имя configure]**. Если имя *configure* не указано, вместо него будет использоваться элемент *USE*. Начиная с **EAPI 4** добавлено распознавание пустого аргумента *опция configure*. В **EAPI 3** и ниже пустая *опция configure* обрабатывается так, как если бы она не была указана.

Примеры:

```
USE="opengl"
myconf=$(use_with opengl)
(новое значение myconf - "--with-opengl")
```

```
USE="jpeg"
myconf=$(use_with jpeg libjpeg)
(новое значение myconf - "--with-libjpeg")
```

```
USE=""
myconf=$(use_with jpeg libjpeg)
(новое значение myconf - "--without-libjpeg")
```

```
USE="sdl"
myconf=$(use_with sdl SDL all-plugins)
(новое значение myconf - "--with-SDL=all-plugins")
```

use_enable [имя configure] [опция configure]

Аналогична описанной выше **use_with**, но с опциями configure **--enable-** вместо **--with-** и **--disable-** вместо **--without-**. Начиная с **EAPI 4** добавлено распознавание пустого аргумента *опция configure*. В **EAPI 3** и ниже пустая *опция configure* обрабатывается так, как если бы она не была указана.

hasv <элемент> <список элементов>

Если элемент входит в список элементов, он выводится и функция **hasv** возвращает 0; в противном случае не выводится ничего, а функция возвращает 1. Как говорилось выше для *use*, существует версия этой функции, которая ничего не выводит, **has**. Используйте **has** во всех случаях, когда вывод информации не нужен. В частности, никогда не используйте вывод для вычислений.

Разделитель списка элементов определяется переменной *IFS*. По умолчанию значение этой переменной - ' ' или пробел. Это переменная **bash(1)**.

has_version <категория/пакет-версия>

Проверяет, установлен ли в системе пакет *категория/пакет-версия*. Параметр может принимать любые значения, допустимые для переменной **DEPEND**. Функция возвращает 0 если пакет, указанный аргументом, установлен, иначе возвращает 1. Начиная с **EAPI 5** здесь можно использовать опцию --host-root, чтобы запрос относился к корню хоста вместо \${ROOT}.

best_version <пакет>

Эта функция ищет *пакет* в текущей базе данных установленных программ и выдает "лучшую версию" устанавливаемого пакета. Начиная с **EAPI 5** здесь можно использовать опцию --host-root, чтобы запрос относился к корню хоста вместо \${ROOT}.

Пример:

```
VERINS=$(best_version net-ftp/glftpd)"
```

(теперь VERINS имеет значение "net-ftp/glftpd-1.27", если установлен glftpd-1.27)

Ловушки

register_die_hook [имена функций]

Привязывает вызов одной или более функций к ошибке ебилда, в том числе в случае конфликта с другими пакетами.

register_success_hook [имена функций]

Привязывает вызов одной или более функций к успешной сборке и/или установке ебилда.

Вывод

einfo "текущее сообщение"

Функция аналогична **elog**, но должна использоваться применительно к сообщениям, не критичным для пользователя (таким как сообщение о ходе сборки или вывод состояния операции).

elog "важная информация"

Если необходимо вывести важное сообщение, которое пользователь обязательно должен прочитать, используйте функцию **elog**. Она работает так же, как **echo(1)**, но предоставляет более подробный вывод, привлекающий внимание пользователя. Кроме того, portage внесет данное сообщение в журнал, чтобы можно было к нему вернуться.

ewarn "предупреждение"

Аналогична **einfo**, но должна использоваться для вывода пользователю предупреждающих сообщений.

eqawarn "предупреждение QA"

Аналогична **einfo**, но должна использоваться для вывода пользователю предупреждающих сообщений по QA.

error "сообщение об ошибке"

Аналогична **einfo**, но должна использоваться для вывода пользователю сообщений об ошибках.

ebegin "полезное сообщение"

Аналогично функции **einfo**, выводит *полезное сообщение* и сообщает о том, что выполнение последующей операции может занять некоторое время. По завершении должна вызываться функция **eend**.

eend <состояние> ["сообщение об ошибке"]

После того, как отработала функция **ebegin**, выводит сообщение, маркированное либо "OK" (при успешном завершении), либо "!!" (в случае ошибки). Если *состояние* не равно нулю, будет выведено дополнительное *сообщение об ошибке*.

Распаковка

unpack <исходный код> [другие исходные коды]

Эта функция распаковывает и/или разархивирует исходный код в текущий каталог. Он будет добавлен в переменную **DISTDIR**.

Компиляция

econf [*опции configure*]

Эта функция используется вместо configure. Она выполняет следующее:

```
 ${ECONF_SOURCE:-.}/configure \
    ${CBUILD:+--build=${CBUILD}} \
    --datadir="${EPREFIX}"/usr/share \
    --host=${CHOST} \
    --infodir="${EPREFIX}"/usr/share/info \
    --localstatedir="${EPREFIX}"/var/lib \
    --prefix="${EPREFIX}"/usr \
    --mandir="${EPREFIX}"/usr/share/man \
    --sysconfdir="${EPREFIX}"/etc \
    ${CTARGET:+--target=${CTARGET}} \
    --disable-dependency-tracking \
    ${EXTRA_ECONF} \
    опции configure || die "econf failed"
```

Обратите внимание, что переменная `_EXTRA_ECONF` отведена для пользователей и не предназначена для написания ebuild-сценариев. Если необходимо передать configure дополнительные опции, сделайте это с помощью **econf**. Заметьте также, что, если сценарий configure выдает ошибку, **econf** автоматически вызывает функцию `die`. Начиная с EAPI 3 функция **econf** обращается к переменной `$_EPREFIX`, которая в предыдущих версиях EAPI не использовалась, а начиная с EAPI 4 - добавляет `--disable-dependency-tracking_` к аргументам, если в выводе команды `configure --help` встречается строка `disable-dependency-tracking`. Начиная с EAPI 5, **econf** добавляет к аргументам `disable-silent-rules`, если в выводе `configure --help` встречается строка `disable-silent-rules_`.

emake [*опции make*]

Эта функция используется вместо make. Она выполняет 'make \${MAKEOPTS} опции make' (согласно настройкам в make.globals); по умолчанию выставлено `MAKEOPTS="-j2"`.

ВНИМАНИЕ

Прежде чем использовать **emake**, убедитесь, что сборка возможна при распараллеливании (`make -j2`). Важно знать это наверняка, поскольку распараллеливание хотя и не всегда, но может приводить к ошибкам. Если при параллельной сборке пакет выдает ошибку, которую не удается разрешить, вместо 'make' воспользуйтесь '**emake -j1**'.

Установка

einstall [*опции make*]

Эта функция используется вместо make install. Она выполняет следующее:

```
make \
    prefix=${ED}/usr \
    datadir=${ED}/usr/share \
    infodir=${ED}/usr/share/info \
    localstatedir=${ED}/var/lib \
    mandir=${ED}/usr/share/man \
    sysconfdir=${ED}/etc \
    ${EXTRA_EINSTALL} \
    опции make \
```

install

Не используйте эту функцию вместо 'emake install DESTDIR=\${D}': данный способ установки предназначен преимущественно для пакетов, собираемых с помощью make. Не используйте также переменную _EXTRA_EINSTALL_, так как она зарезервирована для пользователей.

prepall

prepalldocs

prepallinfo

prepallman

prepallstrip

Эти функции удобны в случае, когда пакет устанавливается в каталог \${D} с помощью сценариев (например, make-файлов). Если необходимо гарантировать, что библиотеки являются выполняемыми, файлы aclocal установлены правильно, все файлы doc/info/man упакованы, из выполняемых файлов удалена отладочная информация, используйте этот набор функций.

prepall:

Запускает функции **prepallman**, **prepallinfo**, **prepallstrip**, устанавливает для библиотек права +x, а затем проверяет каталоги aclocal.

Обратите внимание, что функция **prepalldocs** при этом **не вызывается**.

prepalldocs:

Упаковывает все doc-файлы в каталоге \${ED}/usr/share/doc.

prepallinfo:

Упаковывает все info-файлы в каталоге \${ED}/usr/share/info.

prepallman:

Упаковывает все man-файлы в каталоге \${ED}/usr/share/man.

prepallstrip:

Удаляет отладочную информацию из всех выполняемых файлов - в частности, из библиотек.

prepinfo [каталог]

prepman [каталог]

prepstrip [каталог]

Аналогична функциям **prepall**, но с небольшими различиями.

prepinfo:

Если каталог не указан аргументом, то **prepinfo** принимает за каталог *usr*. Затем **prepinfo** упакует все файлы в каталоге \${ED}/dir/info.

prepman:

Если каталог не указан аргументом, то **prepman** принимает за каталог *usr*. Затем **prepman** упакует все файлы в каталоге \${ED}/dir/man/*/.

prepstrip:

Из всех файлов в каталоге \${ED}/dir удаляется отладочная информация. Можно указывать несколько каталогов.

docompress <путь> [другие пути]

Начиная с **EAPI 4** вспомогательная функция **docompress** используется для управления списками файлов на включение или исключение для необязательной упаковки. Если первый аргумент - **-x**, каждый из последующих аргументов будет добавлен в список на исключение. Без **-x** все аргументы будут добавлены в список на включение. Изначально список на включение содержит каталоги **/usr/share/doc**, **/usr/share/info**, and **/usr/share/man**. Изначальный список на исключение содержит каталог **_usr/share/doc/\$ {PF}/html_**.

Необязательная упаковка выполняется после того, как отработала функция **src_install**, но до того, как начнет выполняться любая последующая фазовая функция. Для каждого элемента списка на включение временно принимается префиксом значение переменной **D**. Далее:

Если это директория, система будет действовать так, как если бы каждый файл или каталог, находящийся непосредственно в этой директории, состоял в списке на включение.

Если это файл, он может быть упакован, если только он не был исключен (см. ниже).

Если элемент не существует, он будет проигнорирован.

Подлежит ли элемент исключению, определяется следующим образом: для каждого элемента списка на исключение временно принимается префиксом значение переменной **D**. Далее:

Если это директория, система будет действовать так, как если бы каждый файл или каталог, находящийся непосредственно в данном каталоге, состоял в списке на исключение.

Если это файл, он не будет упакован.

Если элемент не существует, то он будет проигнорирован.

dosed "s:orig:change:g" <имя файла>

Начиная с **EAPI 4**, вспомогательная функция **dosed** не существует. Ебилды должны вызывать команду **sed(1)** напрямую (принимая, что это sed GNU).

Выполняет команду sed для файла *имя файла* в каталоге \${ED}. Если никакого выражения не задано, то "s:\${D}::g" выступает как выражение по умолчанию. Обратите внимание, что для этого выражения **НЕ ИСПОЛЬЗУЕТСЯ** префикс смещения.

'**dosed\ s:/usr/local:/usr:g \ /usr/bin/some-script'** выполняет команду для файла \${ED}/usr/bin/сценарий.

dodir <путь>

Создает каталог в \${ED}.

'**dodir\ /usr/lib/apache**' создает \${ED}/usr/lib/apache. Обратите внимание, что функции do* будут вызывать функцию **dodir**.

diropts [опции для install(1)]

Может использоваться для задания опций утилиты install, к которым обращается функция **dodir**. Значение по умолчанию - **-m0755**.

into <путь>

Назначает корневой каталог (*DESTTREE*) для других функций, таких как **dobin**, **dosbin**, **doman**, **doinfo**, **dolib**.

Корневой каталог по умолчанию - **/usr**.

keepdir <путь>

Заставляет portage оставить каталог, даже если он пуст. Работает аналогично функции **dodir**.

dobin <бинарный файл> [другие бинарные файлы]

Устанавливает один или несколько бинарных файлов в каталог *DESTTREE/bin*. Функция создает все необходимые каталоги.

dosbin <бинарный файл> [другие бинарные файлы]

Устанавливает один или несколько бинарных файлов в каталог *DESTTREE/sbin*. Функция создает все необходимые каталоги.

doinitd <сценарий init.d> [другие сценарии init.d]

Устанавливает сценарии *init.d* для Gentoo по стандартному для *init.d* пути (*/etc/init.d/*). Функция создает все необходимые каталоги.

doconfd <файл conf.d> [другие файлы conf.d]

Устанавливает файлы *conf.d* для Gentoo по стандартному для *conf.d* пути (*/etc/conf.d/*). Функция создает все необходимые каталоги.

doenvd <запись env.d> [другие записи env.d]

Устанавливает записи *env.d* для Gentoo по стандартному для *env.d* пути (*/etc/env.d/*). Функция создает все необходимые каталоги.

dolib <библиотека> [другие библиотеки]

dolib.a <библиотека> [другие библиотеки]

dolib.so <библиотека> [другие библиотеки]

Устанавливает одну или несколько библиотек в каталог *DESTTREE/lib*. Функция создает все необходимые каталоги.

libopts [опции для *install(1)*]

Может использоваться для задания опций утилите *install*, к которым обращаются функции **dolib**. Значение по умолчанию - *-m0644*.

doman [-i18n=<локаль>] [другие man-файлы]

Устанавливает файлы man-руководств в каталог */usr/share/man/man[0-9]*, в зависимости от суффикса файла. Если это необходимо, файлы будут упакованы. Опция *-i18n* позволяет устанавливать руководства на определенном языке. Затем man-файлы будут установлены в каталог */usr/share/man/<локаль>/man[0-9]*. Начиная с **EAPI 2** man-страницы для определенной локали, в имени которых содержится суффикс локали, устанавливаются в каталог */usr/share/man/<локаль>/man[0-9]*, причем часть имени файла, обозначающая локаль, удаляется, а опция *-i18n* будет проигнорирована. Например, если используется **EAPI 2**, man-страница с именем *foo.<локаль>.1* будет установлена как */usr/share/man/<локаль>/man1/foo.1*. Начиная с **EAPI 4** атрибут опции *-i18n* имеет приоритет перед суффиксом локали в имени файла.

dohard <имя файла> <имя ссылки>

Начиная с **EAPI 4** вспомогательная функция **dohard** не существует. Ебилды должны вызывать утилиту **ln(1)** напрямую.

dosym <имя файла> <имя ссылки>

Выполняет команду **ln**, создавая тем самым символьическую ссылку.

doheader <имя файла> [другие файлы]

Устанавливает данные файлы заголовков в каталог */usr/include/* с правами по умолчанию *0644* (можно переопределить через функцию **insopts**). Опция **-g** включит в обработку содержимое подкаталогов. Вспомогательная функция **doheader** доступна начиная с **EAPI 5**.

dohtml _ [-a типы-файлов] [-r] [-x игнорируемые-каталоги] [файлы-и-каталоги]_

Устанавливает файлы из списка файлов (разделяются пробелом) в каталог */usr/share/doc/\${PF}/html*, если файлы имеют расширение *.htm*, *.html*, *.css*, *.js*, *.gif*, *.jpeg*, *.jpg* или *.png*. Опция **-a** позволяет ограничить действие определенными типами файлов, опция **-A** добавляет типы к перечню по умолчанию, **-x** определяет каталоги, которые необходимо пропустить (по умолчанию пропускается **CVS**), **-r** определяет префикс документа, с опцией **-r** каталоги обрабатываются рекурсивно.

doinfo [другие info-файлы]

Устанавливает info-файлы в каталог *DESTDIR/info*. Все они автоматически архивируются с помощью **gzip**. Функция создает все необходимые каталоги.

domo _<файл локали> [другие файлы локали] _

Устанавливает файлы локали в каталог *DESTDIR/usr/share/locale/[ЯЗЫК]*, исходя из их суффикса. Функция создает все необходимые каталоги.

fowners <права доступа> <файл> [файлы]

fperms <права доступа> <файл> [файлы]

Выполняет chown (**fowners**) или chmod (**fperms**), назначая тем самым *права доступа к файлам*.

insinto [путь]

Устанавливает корневой каталог для функции **doins**.

По умолчанию им является /.

insopts [опции для *install(1)*]

Может быть использована для задания опций утилите *install*, используемой в функции **doins**. Значение по умолчанию - *-m0644*.

doins <файл> [другие файлы]

Устанавливает файлы по пути, который управляется функцией **insinto**. Эта функция использует утилиту *install(1)*. Все необходимые каталоги будут созданы. Опция -r устанавливает рекурсивный режим.

Начиная с **EAPI 4** и **doins**, и **newins** сохраняют символические ссылки, тогда как в **EAPI 3** и более ранних версиях символические ссылки не сохранялись, а происходило разыменование.

exeinto [путь]

Устанавливает корневой каталог для функции **doexe**.

По умолчанию им является /.

exeopts [опции для *install(1)*]

Может использоваться для задания опций утилите *install*, используемых в функции **doexe**. Значение по умолчанию - *-m0755*.

doexe <исполняемый файл> [другие исполняемые файлы]

Устанавливает исполняемые файлы в каталог, управляемый **exeinto**. Эта функция использует утилиту *install(1)*. Функция создает все необходимые каталоги.

docinto [путь]

Назначает подкаталог, используемый функциями **dodoc** и **dohtml** при установке в дерево документов (на основе */usr/share/doc/\${PF}/*). По умолчанию подкаталога либо нет, либо установлено значение "".

dodoc <документ> [другие документы]

Устанавливает один или несколько документов в каталог */usr/share/doc/\${PF}/<путь docinto>*.

Документы помечаются для упаковки. Функция создает все необходимые каталоги. Начиная с **EAPI 4** поддерживается рекурсивная обработка подкаталогов; чтобы включить ее, используйте опцию *-r*.

newbin <прежний файл> <новое имя файла>

newsbin <прежний файл> <новое имя файла>

newinitd <прежний файл> <новое имя файла>

newconfd <прежний файл> <новое имя файла>

newenvd <прежний файл> <новое имя файла>

newlib.so <прежний файл> <новое имя файла>

newlib.a <прежний файл> <новое имя файла>

newman <прежний файл> <новое имя файла>

newinfo <прежний файл> <новое имя файла>

newins <прежний файл> <новое имя файла>

newexe <прежний файл> <новое имя файла>

newdoc <прежний файл> <новое имя файла>

Все эти функции работают аналогично функциям **do**, но только с одним файлом, который устанавливается с [новым именем]. Начиная с **EAPI 5**, если первый параметр - дефис (-), происходит обращение к стандартному вводу.

ПРИМЕРЫ

```
# Copyright 1999-2009 Gentoo Foundation
# Distributed under the terms of the GNU General Public License v2
# $Header: $

EAPI="5"

inherit some_eclass another_eclass

DESCRIPTION="Super-useful stream editor (sed)"
HOMEPAGE="http://www.gnu.org/software/sed/sed.html"
SRC_URI="ftp://alpha.gnu.org/pub/gnu/${PN}/${P}.tar.gz"

LICENSE="GPL-2"
SLOT="0"
KEYWORDS="~x86"
IUSE=""

RDEPEND=""
DEPEND="nl? ( sys-devel/gettext )

src_configure() {
    econf
        --bindir="${EPREFIX}/bin
}

src_install() {
    emake DESTDIR="${D}" install
    dodoc NEWS README* THANKS AUTHORS BUGS ChangeLog
}
```

ФАЙЛЫ

Сценарий */usr/sbin/ebuild.sh*.

Вспомогательные приложения в каталоге */usr/lib/portage/bin*.

/etc/make.conf

Содержит переменные для процесса сборки; они имеют приоритет над заданными в *make.defaults*.

/usr/share/portage/config/make.globals

Содержит переменные по умолчанию для процесса сборки. Редактировать значения следует не здесь, а в файле */etc/make.conf*.

/etc/portage/color.map

Содержит переменные для пользовательских настроек цветного вывода.

СМ. ТАКЖЕ

[ebuild](#) (1), [make.conf](#) (5), [color.map](#) (5)

БАГТРЕКЕР

Об обнаруженных ошибках сообщайте на <http://bugs.gentoo.org/>.

АВТОРЫ

- Achim Gottinger [\[achim@gentoo.org\]](mailto:achim@gentoo.org)(mailto:achim@gentoo.org)\\
- Mark Guertin [\[gerk@gentoo.org\]](mailto:gerk@gentoo.org)(mailto:gerk@gentoo.org)\\
- Nicholas Jones [\[carpaski@gentoo.org\]](mailto:carpaski@gentoo.org)(mailto:carpaski@gentoo.org)\\
- Mike Frysinger [\[vapier@gentoo.org\]](mailto:vapier@gentoo.org)(mailto:vapier@gentoo.org)\\
- Arfrever Frethes Taifersar Arahesis [\[Arfrever.FTA@gmail.com\]](mailto:Arfrever.FTA@gmail.com)(mailto:Arfrever.FTA@gmail.com)\\
- Fabian Groffen [\[grobian@gentoo.org\]](mailto:grobian@gentoo.org)(mailto:grobian@gentoo.org)\\

ПЕРЕВОД

- Елена Гаврилова [\[e.vl.gavrilova@yandex.ru\]](mailto:e.vl.gavrilova@yandex.ru)(mailto:e.vl.gavrilova@yandex.ru)\\

Октябрь 2012

EGENCACHE

НАЗВАНИЕ

egencache - кэширование метаданных репозитариев eбилдов

СИНТАКСИС

egencache [опции] --update [ATOM]...

ОПИСАНИЕ

Утилита *egencache* кэширует метаданные репозитариев eбилдов и сохраняет их в каталоге *metadata/cache/* внутри самого репозитария, для дальнейшего распространения.

ДЕЙСТВИЯ

--update [ATOM] ...

Обновляет каталог *metadata/cache/* (метаданные будут генерироваться по необходимости). Если атомы пакетов не указаны, обновлены будут все элементы подряд. Подробнее о синтаксисе атомов пакетов см. [ebuild\(5\)](#).

--update-changelogs

Обновляет ChangeLog-файлы из журналов SCM (поддерживается только для репозитариев на Git).

--update-use-local-desc

Обновляет файл *profiles/use.local.desc* из *metadata.xml*.

ОПЦИИ

--cache-dir=CACHE_DIR

Определяет расположение промежуточного кэша метаданных, который сохраняется в другом формате, включающем состояние еклассов. О том, почему в этом необходимо, см. в разделе **ОШИБКИ**.

По умолчанию - */var/cache/edb/dep*.

--config-root=PORTAGE_CONFIGROOT

Определяет расположение конфигурационных файлов portage.

По умолчанию это корневой каталог, */*.

--ignore-default-opt

При использовании этой опции значения *_EGENCACHE_DEFAULT_OPTS_* будут игнорироваться.

--jobs=JOBS

Определяет максимальное число распараллеленных процессов сборки. См. также родственную опцию --load-average.

--load-average=LOAD

Определяет максимальную среднюю загрузку при параллельной сборке.

--portdir=PORTDIR

Переопределяет расположение дерева портежей.

--portdir-overlay=PORTDIR_OVERLAY

Переопределяет переменную PORTDIR_OVERLAY (одновременно необходимо использовать опцию --repo).

--preserve-comments

Сохраняет комментарии, обнаруженные в файле вывода use.local.desc. Для этого файл вывода должен существовать до вызова egencache.

--repo=REPO

Определяет имя рабочего репозитария (по умолчанию репозитарий расположен в PORTDIR). Оно должно соответствовать записи **repo_name** (см. [portage\(5\)](#)) одного из репозитариев, сконфигурированных с помощью переменной PORTDIR или PORTDIR_OVERLAY (см. [make.conf\(5\)](#)).

--rsync

При использовании в связке с действием --update включает альтернативный способ синхронизации на тот случай, если содержимое кэша изменилось, но не изменилось ни время изменения, ни размер, вследствие чего rsync не может отследить изменения; подобные ситуации разрешаются путем обновления времени последнего изменения (и соответствующей записи в кэше). Эта опция имеет смысл только при распространении данных с помощью таких механизмов как rsync(1), осуществляющих контроль версий на основе меток времени и размера файлов (см. [ошибка 139134](#)). Она не нужна, если используется git(1), поскольку последний работает более тонко, выявляя изменения в инодах (описание см. в файле racy-git.txt из комплекта документации по Git).

--tolerant

Сообщить об успешном завершении и выйти в случае незначительных ошибок, таких как пропуск обновления кэша, когда ebuild либо не удалось использовать в качестве источника данных, либо он не был загружен сам по причине некорректного Manifest-файла.

--use-local-desc-output=ULD_OUTPUT

файл вывода данных use.local.desc (или '-' для стандартного вывода)

ОПЦИИ ОКРУЖЕНИЯ

EGENCACHE_DEFAULT_OPTS

Если эта переменная установлена в [make.conf\(5\)](#), то все опции, которые она содержит, будут добавлены в начало командной строки при каждом вызове. Временно это можно отменить использованием опции --ignore-default-opts.

ОШИБКИ

Формат кэшированных метаданных, хранящихся и доступных для получения в каталоге репозитария *metadata/cache/*, налагает серьезные ограничения - прежде всего, на механизм проверки. Последний, в настоящее время, основан на сопоставлении времени последнего изменения каждой записи в кэше с временем изменения соответствующего [ebuild](#)-сценария. В связи с тем, что кэш не содержит информации о состоянии еклассов, в случае, когда изменение еклассов влечет за собой изменение метаданных, этот механизм ненадежен. Кроме того, поскольку время изменения кэша должно совпадать

со временем изменения ебилда, формат кэша пригоден лишь для передачи данных по протоколам, поддерживающим сохранение отметок времени (например, посредством **rsync(1)**). Если кэшированные данные находятся в **git(1)**-репозитариях, проблему возможно обойти с помощью команды **emerge --sync**: она обновит время изменения ебилдов согласно записям в кэше (это не относится лишь к тем ебилдам, которые были изменены по HEAD).

Для разрешения этих проблем в дальнейшем планируется введение расширенного формата кэша, который включит в себя дополнительные проверочные данные в виде digest-файлов как для ебилда, так и для наследуемых им еклассов. Пока этого не произошло, вам следует выставить **metadata-transfer** в переменной **FEATURES** (см. **make.conf(5)**). В результате будет генерироваться промежуточный кэш (в ином формате, включающем состояние еклассов); его рабочий каталог можно настроить, прибегнув к опции **--cache-dir**.

БАГТРЕКЕР

Об обнаруженных ошибках сообщайте на <http://bugs.gentoo.org/>

АВТОРЫ

- Zac Medico [\[zmedico@gentoo.org\]\(mailto:zmedico@gentoo.org\)\](mailto:zmedico@gentoo.org)

ФАЙЛЫ

/etc/make.conf

Содержит переменные.

СМ. ТАКЖЕ

[emerge\(1\)](#), [make.conf\(5\)](#), [portage\(5\)](#)

ПЕРЕВОД

- Елена Гаврилова [\[e.vl.gavrilova@yandex.ru\]\(mailto:e.vl.gavrilova@yandex.ru\)\](mailto:e.vl.gavrilova@yandex.ru)

Октябрь 2010

EMAINT

НАЗВАНИЕ

emaint - проверка состояния системы и обеспечение ее работоспособности

СИНТАКСИС

emaint [*опции*] [**all** | **binhost** | **cleanresume** | **movebin** | **moveinst** | **world**]

ОПИСАНИЕ

Утилита **emaint** предоставляет интерфейс для проверки целостности системы и обеспечения ее работоспособности.

КОМАНДЫ

all

Выполняет все поддерживаемые команды.

binhost

Генерирует индекс метаданных для бинарных пакетов, хранящихся в каталоге **PKGDIR** (для удаленного скачивания). См. документацию по **PORTAGE_BINHOST** в тап-руководстве по **make.conf(5)**.

cleanresume

Очищает списки устанавливаемых пакетов, сохраненные для операции [**emerge --resume**](#).

movebin

Обновляет местонахождение бинарных пакетов из каталога **PKGDIR**.

moveinst

Обновляет местонахождение установленных пакетов.

world

Исправляет ошибки в файле *world*.

ОПЦИИ

-c, --check

Проверить на возможные проблемы.

-f, --fix

Исправить все возможные проблемы.

БАГТРЕКЕР

Об обнаруженных ошибках сообщайте на <http://bugs.gentoo.org/>

АВТОРЫ

- Mike Frysinger [\[vapier@gentoo.org\]\(mailto:vapier@gentoo.org\)\](mailto:vapier@gentoo.org)

ФАЙЛЫ

/var/lib/portage/world

Содержит перечень всех пакетов, определенных пользователем.

СМ. ТАКЖЕ

[**emerge\(1\)**](#), [**portage\(5\)**](#)

ПЕРЕВОД

- Елена Гаврилова [\[e.vl.gavrilova@yandex.ru\]\(mailto:e.vl.gavrilova@yandex.ru\)\](mailto:e.vl.gavrilova@yandex.ru)

Ноябрь 2008

EMERGE

НАЗВАНИЕ

emerge - консольный интерфейс к системе портежей

СИНТАКСИС

emerge

[*опции*] [*действие*] [*ебилд* | *архив tbz2* | *файл* | *@сем* | *атом*] ...

emerge

--sync | **--version**

emerge

--info [*атом*]

emerge

--search *строка*

emerge

--help [--verbose]

ОПИСАНИЕ

emerge - это полный интерфейс командной строки к системе портежей. Он используется прежде всего для установки пакетов, причем **emerge** умеет автоматически обрабатывать любые зависимости пакета. **emerge** может также обновлять **дерево портежей**, делая новые и обновленные пакеты доступными для установки. Утилита обеспечивает беспроблемное обновление уже установленных пакетов до новых версий. Она работает как с исходными кодами, так и с бинарными пакетами, а также может быть использована для создания собственных бинарных пакетов для дальнейшего распространения.

ЕБИЛДЫ, АРХИВЫ, СЕТЫ И АТОМЫ

Основное назначение **emerge** - установка пакетов. Пакеты можно задать одним из следующих способов: как *атом*, как *сет*, как установленный *файл*, как *ебилд* или как архив *tbz2*.

ебилд

Ебилд должен быть задан, как минимум, в виде допустимого имени каталога пакетов Portage, как без указания версии или категории, например, **portage** или **python**, так и с добавлением таковых, например, **sys-apps/portage** или **=python-2.2.1-r2**. **emerge** игнорирует завершающую косую черту, поэтому можно использовать автоматическое завершение имени файла. Значением *ебилда* может быть и фактическое имя файла, например, **/usr/portage/app-admin/python/python-2.2.1-r2.ebuild**. **ВНИМАНИЕ:** на данный момент синтаксис **emerge /путь/к/ебилду** не работает и не может быть использован.

архив tbz2

Архив *tbz2* должен быть корректным файлом с расширением *.tbz2*, созданным с помощью команды **ebuild <пакет>-<версия>.ebuild package**, **emerge --buildpkg [категория/]<пакет>** или **quickpkg /var/db/pkg/<категория>/<пакет>**.

файл

Файл должен быть файлом или каталогом, установленным одним или несколькими пакетами. Если не используется абсолютный путь, файл в аргументе следует задавать с префиксом **./** или **../**. Для каталогов, которые принадлежат нескольким пакетам, должны быть выбраны все эти пакеты. Если вы хотите запросить владельца одного или нескольких файлов или каталогов, см. команду **portageq(1)**.

сет

Сет - удобный способ обозначить большую группу пакетов. В настоящее время поддерживаются три сета пакетов, которые доступны всегда: **selected**, **system** и **world**. Сет **selected** содержит определенные пользователем **world**-пакеты, которые перечислены в файле **/var/lib/portage/world**, и вложенные сети, которые могут быть указаны в файле **/var/lib/portage/world_sets**. Сет **system** содержит набор пакетов, которые считаются необходимыми для правильной работы системы. Сет **world** включает пакеты как из **selected**, так и из **system**. [Подробнее см. ниже в разделе **ФАЙЛЫ**.] В зависимости от конфигурации системы, могут быть доступны и другие сети. Конфигурация сетов по умолчанию находится в каталоге **/usr/share/portage/config/sets**. Обратите внимание, что *сет* в аргументе **emerge** используется, как правило, при выполнении действия **--update**. Для корректного распознавания сети должны иметь префикс **@**. Если вы хотите просмотреть список доступных сетов, используйте действие **--list-sets**.

атом

Атом описывает ограничения на пакет, который необходимо установить. *Подробнее о синтаксисе атомов* см. **ebuild(5)**. Например, **>=dev-lang/python-2.2.1-r2** соответствует последней доступной версии Python не ниже 2.2.1-r2. Аналогично, **<dev-lang/python-2.0** соответствует последней доступной версии Python ниже 2.0. Учтите, что во многих командных интерпретаторах такие символы как '**'**

БАГТРЕКЕР

Сообщайте нам обо всех обнаруженных ошибках: <http://bugs.gentoo.org/>. Не забудьте включить в сообщение вывод команды **emerge --info**.

АВТОРЫ

- Daniel Robbins [drobbins@gentoo.org](mailto:drobbins@gentoo.org)\\
- Geert Bevin [gbevin@gentoo.org](mailto:gbevin@gentoo.org)\\
- Achim Gottinger [achim@gentoo.org](mailto:achim@gentoo.org)\\
- Nicholas Jones [carpaski@gentoo.org](mailto:carpaski@gentoo.org)\\
- Phil Bordelon [phil@thenexusproject.org](mailto:phil@thenexusproject.org)\\
- Mike Frysinger [vapier@gentoo.org](mailto:vapier@gentoo.org)\\
- Marius Mauch [genone@gentoo.org](mailto:genone@gentoo.org)\\
- Jason Stubbs [jstubbs@gentoo.org](mailto:jstubbs@gentoo.org)\\
- Brian Harring [ferringb@gmail.com](mailto:ferringb@gmail.com)\\
- Zac Medico [zmedico@gentoo.org](mailto:zmedico@gentoo.org)\\

ФАЙЛЫ

Здесь приведены основные файлы, отвечающие за работу emerge. Если вас интересует полный перечень, обратитесь к man-руководству **portage(5)**.

/usr/share/portage/config/sets/

Содержит стандартные конфигурации сетов.

/var/lib/portage/world

Содержит список всех указанных пользователем пакетов. Вы можете свободно редактировать этот файл, добавляя пакеты, которые необходимо учитывать при обновлении сета **world**, и удаляя те, которые обновлять при этом не нужно.

/etc/make.conf

Содержит переменные сборки, имеющие приоритет над значениями, указанными в **make.globals**.

/etc/portage/color.map

Содержит переменные, используемые для пользовательской настройки цветного вывода.

/etc/dispatch-conf.conf

Содержит настройки автоматического обновления/отката конфигурационных файлов.

/etc/make.profile/make.defaults

Содержит специфические для выбранного профиля переменные сборки. **Не редактируйте этот файл.**

/usr/portage/profiles/use.desc

Содержит основной список USE-флагов с описаниями их функций. **Не редактируйте этот файл.**

/etc/make.profile/virtuals

Содержит стандартный список пакетов, используемый для разрешения виртуальных зависимостей. **Не редактируйте этот файл.**

/etc/make.profile/packages

Содержит список пакетов для базовой системы. Сеты **system** и **world** обращаются именно к нему. **Не редактируйте этот файл.**

/usr/share/portage/config/make.globals

Содержит переменные по умолчанию по сборки. **Не редактируйте этот файл.**

СМ. ТАКЖЕ

emerge --help , **quickpkg** (1), **ebuild** (1), **ebuild** (5), **make.conf** (5), **color.map** (5), **portage** (5) Ряд вспомогательных приложений расположены в каталоге **/usr/lib/portage/bin**, а пакет **app-portage/gentoolkit** содержит полезные сценарии, в частности, **equery** (утилита для запроса информации о пакетах).

ПЕРЕВОД

- Елена Гаврилова [e.vl.gavrilova@yandex.ru](mailto:e.vl.gavrilova@yandex.ru)\

Октябрь 2011

ENV-UPDATE

НАЗВАНИЕ

env-update - автоматическое обновление настроек окружения

СИНТАКСИС

env-update [опции]

ОПИСАНИЕ

env-update обрабатывает файлы в каталоге */etc/env.d* и автоматически генерирует */etc/profile.env* и */etc/ld.so.conf*. Затем для обновления */etc/ld.so.cache* запускается **ldconfig(8)**. **emerge(1)** автоматически вызывает **env-update** после каждой установки пакета. Если же вы вносите изменения в */etc/env.d*, вам следует самостоятельно выполнить *env-update*, чтобы внесенные изменения вступили в силу. Обратите внимание, что это повлияет только на последующие операции. Чтобы изменения отразились на уже запущенных процессах, вероятно, понадобится выполнить *source /etc/profile*.

ОПЦИИ

--no-ldconfig

Не запускать ldconfig (и, тем самым, опустить пересборку кэша ld.so cache и т.д.).

БАГТРЕКЕР

Об обнаруженных ошибках сообщайте на <http://bugs.gentoo.org/>

АВТОРЫ

Daniel Robbins [drobbins@gentoo.org](mailto:drobbins@gentoo.org)\

СМ. ТАКЖЕ

[emerge\(1\)](#), [ldconfig\(8\)](#)

ПЕРЕВОД

- Елена Гаврилова [e.vl.gavrilova@yandex.ru](mailto:e.vl.gavrilova@yandex.ru)\

Август 2008

ETC-UPDATE

НАЗВАНИЕ

etc-update - обработка изменений конфигурационных файлов

СИНТАКСИС

etc-update

ОПИСАНИЕ

Утилиту *etc-update* следует запускать после установки новых пакетов для проверки предлагаемых обновлений конфигурационных файлов. Если новый конфигурационный файл может перезаписать имеющийся, *etc-update* спросит у пользователя, как с ним поступить.

etc-update проверяет все каталоги из переменной *_CONFIG_PROTECT_*, а конфигурационные файлы из *_CONFIG_PROTECT_MASK_* обновляются автоматически. Подробнее об этом см. в справке по [make.conf\(5\)](#).

ОПЦИИ

Нет.

БАГТРЕКЕР

Об обнаруженных ошибках сообщайте на <http://bugs.gentoo.org/>

АВТОРЫ

- Jochem Kossen and Leo Lipelis
- Karl Trygve Kalleberg [\[mailto:karltk@gentoo.org\]](mailto:karltk@gentoo.org)
- Mike Frysinger [\[mailto:vapier@gentoo.org\]](mailto:vapier@gentoo.org)

ФАЙЛЫ

/etc/etc-update.conf

Здесь хранятся настройки *etc-update*.

/etc/dispatch-conf.conf

Настройки для обработки конфигурационных файлов перед их изменением (с помощью *dispatch-conf*).

СМ. ТАКЖЕ

[make.conf\(5\)](#)

ПЕРЕВОД

- Елена Гаврилова [\[mailto:e.vl.gavrilova@yandex.ru\]](mailto:e.vl.gavrilova@yandex.ru)

Август 2008

MAKE.CONF

НАЗВАНИЕ

make.conf - пользовательские настройки Portage

ПАРАМЕТРЫ

/etc/make.conf и /etc/portage/make.conf

DESCRIPTION

Этот файл содержит различные переменные, используемые Portage. В поиске настроек Portage в первую очередь проверяет текущие значения переменных окружения; если они не содержат искомых настроек, Portage обращается к файлам *make.conf*. При этом будет проверен как */etc/make.conf*, так и */etc/portage/make.conf* (если присутствует в системе), причем настройки, содержащиеся в */etc/portage/make.conf*, имеют приоритет над настройками из */etc/make.conf*. Если и обращение к файлам *make.conf* оказалось безрезультатным, Portage проверяет *make.globals*. Последним по порядку обращения стоит файл */etc/make.profile/make.defaults*. Заметьте, что любые пользовательские настройки следует либо конфигурировать в окружении, либо вносить в файлы *make.conf* - именно они

предназначены для редактирования пользователем.

Исключение составляют инкрементные переменные USE, CONFIG_PROTECT*, и ACCEPT_KEYWORDS. Инкрементные переменные обрабатываются в обратном порядке: make.defaults - make.globals - make.conf - конфигурация окружения. Обнуление этих переменных требует полной инициализации: export USE="-*"

Если вы хотите сопоставить пакетам различные настройки, см. раздел **package.env** в руководстве по [portage\(5\)](#).

ПЕРЕМЕННЫЕ

ACCEPT_CHOSTS = [значения CHOST, через пробел]

Содержит приемлемые значения **CHOST**. Поддерживается синтаксис регулярных выражений, поэтому необходимо использовать управляющие знаки для символов **CHOST**, имеющих особое значение в регулярных выражениях.

По умолчанию принимает значение \$CHOST.

ACCEPT_KEYWORDS = [KEYWORDS, через пробел]

Разрешает установку для тестирования ебилдов, которые пока не имеют статуса стабильных. Если архитектура вашей системы x86, установите значение '~x86', если же вы пользователь ppc, значение будет '~ppc'. Это инкрементная переменная; определяется только как ~arch.

По умолчанию принимает значение \$ARCH.

ACCEPT_LICENSE = [список лицензий или групп, через пробел]

Эта переменная используется для маскировки пакетов по лицензии. Она может содержать имена как лицензии, так и группы, причем имена групп имеют префикс @. Группы лицензий определяются в файле _license_groups (см. [portage\(5\)](#)). Помимо имен лицензий и групп лицензий, поддерживаются также подстановочные знаки * и -*. Подробности см. в GLEP 23:

<http://www.gentoo.org/proj/en/glep/glep-0023.html>.

По умолчанию принимает значение * -@EULA. Примеры:

```
# Принимать любую лицензию
ACCEPT_LICENSE="*"
# Принимать любую лицензию, кроме public-domain"
ACCEPT_LICENSE="* -public-domain"
# Принимать лицензию только из группы FSF-APPROVED
ACCEPT_LICENSE="- * @FSF-APPROVED"
```

ACCEPT_PROPERTIES = [параметры, через пробел]

Эта переменная используется для маскировки пакетов по PROPERTIES. Помимо собственно названий параметров, можно использовать подстановочные знаки * и -*. Временно изменять стандартное значение этой переменной можно, используя опцию --accept-properties при запуске [emerge\(1\)](#).

Подробнее о PROPERTIES см. в руководстве по [ebuild\(5\)](#).

По умолчанию принимает значение *.

Примеры:

```
# Принимать любые параметры
ACCEPT_PROPERTIES="*"
# Принимать любые параметры, кроме interactive
ACCEPT_PROPERTIES="* -interactive"
```

CBUILD

Эта переменная передается сценариями ebuild действию configure как --build=\${CBUILD} только в том случае, если она определена. Не редактируйте ее значение, если не вполне уверены в том, что делаете.

CCACHE_DIR = [путь]

Определяет расположение рабочего каталога ccache. Подробнее см. в man-руководстве по **ccache(1)**. По умолчанию это каталог /var/tmp/ccache.

CCACHE_SIZE = "размер"

Ограничивает объем пространства, используемого ccache. По умолчанию - 2 гигабайта ('2G'). В качестве величины измерения можно использовать гигабайты ('G'), мегабайты ('M') и килобайты ('K').

CFLAGS, CXXFLAGS

Используйте эти переменные, чтобы выставить по вашему желанию настройки оптимизации/инструкции процессору для компилируемых приложений. Данные переменные передаются соответственно компиляторам C и C++. (CXX обозначает компилятор C++ на многих системах сборки.) Файлы ебилдов практически всегда учитывают пользовательские настройки, благодаря чему ваша система Gentoo Linux полностью оптимизирована под ваши требования. Тем не менее советуем не экспериментировать с ненадежными настройками в погоне за оптимизацией: их использование может привести к тому, что отдельные пакеты не скомпилируются или не будут работать.

Почитать об этом подробнее можно в разделе *Invoking GCC* ("Вызов GCC") man-руководства по gcc:
<http://gcc.gnu.org/onlinedocs/>

CHOST

Эта переменная передается *сценариями ebuild* действию *configure* как --host=\${CHOST}. Тем самым вы можете принудительно вызвать хост сборки.

Дополнительно см.:

<http://gcc.gnu.org/onlinedocs/gcc-4.1.1/gcc/Submodel-Options.html>
<http://gcc.gnu.org/onlinedocs/gcc-3.3/gcc/Submodel-Options.html>
<http://gcc.gnu.org/onlinedocs/gcc-3.2/gcc/Submodel-Options.html>
http://gcc.gnu.org/onlinedocs/gcc-2.95.3/gcc_2.html

CLEAN_DELAY = целое число

Определяет продолжительность обратного отсчета после запуска `emerge --unmerge`, прежде чем начнется удаление.

По умолчанию - 5 секунд.

COLLISION_IGNORE = [файлы и/или каталоги, через пробел]

Через эту переменную пользователь может отключить *collision-protect* и *protect-owned* для отдельных файлов и/или каталогов.

По умолчанию они отключены только для /lib/modules.

CONFIG_PROTECT = [файлы и/или каталоги, через пробел]

Ко всем указанным здесь файлам и/или каталогам будет применена т.н. "защита конфигурационных файлов". Подробнее об этом см. в разделе **КОНФИГУРАЦИОННЫЕ ФАЙЛЫ** руководства по **emerge(1)**.

CONFIG_PROTECT_MASK = [файлы и/или каталоги, через пробел]

Для всех указанных здесь файлов и/или каталогов защита конфигурационных файлов будет отключена. Подробнее об этом см. в разделе **КОНФИГУРАЦИОННЫЕ ФАЙЛЫ** руководства по **emerge(1)**.

CTARGET

Эта переменная передается *сценариями ebuild* действию *configure* как --target=\${CTARGET} только если она определена.

DISTDIR = [путь]

Определяет расположение вашего локального репозитория исходных кодов. После того, как пакеты собраны, можно безопасно удалить любые файлы из этого каталога, так как они будут загружены автоматически по требованию для данной сборки. Чтобы удалить файлы из каталога выборочно, см. **eclean(1)** из пакета gentoolkit.

Используйте переменную **PORTAGE_RO_DISTDIRS**, чтобы определить один или несколько каталогов только для чтения, содержащих исходные коды.

Имейте в виду, что размещение данных в каталоге /usr/portage может оказаться небезопасным. Более подробную информацию об этом вы найдете в описании переменной **PORTDIR**. По умолчанию это каталог /usr/portage/distfiles.

DOC_SYMLINKS_DIR

Если эта переменная содержит каталог, символьская ссылка на html-документацию будет помещена в него.

EVEEP_IGNORE

Определяет, следует ли игнорировать звуковые сигналы при отображении важных информационных сообщений. По умолчанию эта переменная сброшена.

EMERGE_DEFAULT_OPTS

Позволяет указать опции, которые будут добавлены к каждому вызову [emerge\(1\)](#) в командной строке. Если определено значение --ignore-default-opts, эти опции добавляться не будут.

EMERGE_LOG_DIR

Управляет расположением emerge.log и emerge-fetch.log.

По умолчанию они расположены в каталоге /var/log.

EMERGE_WARNING_DELAY = целое число

Определяет продолжительность обратного отсчета после запуска `emerge --unmerge`, прежде чем начнется удаление системных пакетов.

Время по умолчанию - 10 секунд.

EPAUSE_IGNORE

Определяет, следует ли игнорировать краткие паузы, возникающие при отображении важных информационных сообщений. По умолчанию эта переменная сброшена. Установка ее на любое значение вызовет игнорирование пауз.

EXTRA_ECONF = [строка опций configure]

Содержит дополнительные опции, которые **econf** добавит к аргументам сценария configure (см. [ebuild\(5\)](#)).

FEATURES = "sandbox"

Определяет стандартные действия portage. Это инкрементная переменная. Большинство этих настроек предназначены для разработчиков, но некоторые могут быть полезны и обычному пользователю. Функция **sandbox** очень важна и не должна быть отключена по умолчанию.

assume-digests

При коммитах в cvs с использованием [repoman\(1\)](#) принимать, что все существующие обзоры SRC_URI корректны. Эта функция также влияет на то, как генерируют обзоры [ebuild\(1\)](#) и [emerge\(1\)](#) (emerge делает это только при условии, что включена функция *digest*). Для файлов, отсутствующих в \${DISTDIR}, существующие обзоры принимаются автоматически, даже если функция **assume-digests** отключена. Если же файл присутствует в \${DISTDIR}, но его размер не соответствует существующему обзору, обзор будет создан заново, независимо от того, включена ли **assume-digests**. Команда [ebuild\(1\)](#) **digest** имеет опцию **--force**, которая может использоваться для принудительного пересоздания обзоров.

binpkg-logs

Сохранять журналы успешной установки бинарных пакетов. Это имеет смысл только в том случае, если установлена переменная **PORT_LOGDIR**.

buildpkg

Для всех устанавливаемых пакетов будут создаваться бинарные. См. также [quickpkg\(1\)](#) и [emerge\(1\)](#), опции **--buildpkg** и **--buildpkgonly**.

buildsyspkg

Бинарные пакеты будут создаваться только для сета system.

candy

Использовать особый индикатор выполнения действия, когда [emerge\(1\)](#) ищет зависимости.

ccache

Включить поддержку пакета ccache для portage. Если каталога ccache нет в пользовательской среде, то portage будет по умолчанию использовать \${PORTAGE_TMPDIR}/ccache.

Внимание: Как известно, эта функция может вызывать множественные отказы компиляции. Иногда ccache сохраняет устаревшие объекты кода или поврежденные файлы, что может привести к невозможности установить пакет. В таком случае (если вы получаете сообщение об ошибке типа "File not recognized: File truncated" ("Файл не распознан: усеченный файл")), сначала попробуйте перекомпилировать приложение с отключенным ccache и только потом обращайтесь на багтрекер. Если вы не занимаетесь разработкой, не включайте ccache.

clean-logs

Автоматически выполнять команду, назначенную переменной PORT_LOGDIR_CLEAN. По умолчанию PORT_LOGDIR_CLEAN удалит все файлы из каталога PORT_LOGDIR, которые были изменены за последние 7 дней.

collision-protect

QA-Функция, которая гарантирует, что пакет не перезапишет файлы, которые ему не принадлежат. Для выборочного отключения ее можно использовать переменную COLLISION_IGNORE. См. также связанную функцию *protect-owned*.

compress-build-logs

Все журналы сборки будут сжаты в процессе записи; их имена будут иметь соответствующие типу сжатия расширения. В настоящее время поддерживается только сжатие с помощью [gzip\(1\)](#), соответственно, если данная опция включена, журналы будут иметь расширение '.gz'.

digest

При вызове команд [emerge\(1\)](#), [ebuild\(1\)](#) или [repoman\(1\)](#) будут автоматически создаваться обзоры пакетов. Если включена также функция *assume-digests*, существующие обзоры SRC_URI будут использованы всякий раз, когда они доступны.

distcc

Включить поддержку distcc пакета для portage.

distcc-pump

Включить поддержку distcc пакета для portage, с римп-режимом.

distlocks

Portage будет использовать lockfiles, чтобы гарантировать, что конкурирующие экземпляры не затрут файлы друг друга. Эта функция включена по умолчанию, но может вызвать проблемы на менее интеллектуальных удаленных файловых системах, например, NFSv2, и некоторых специфически конфигурированных Samba-серверах (oplocks off, NFS re-export). Их можно разрешить, используя утилиту /usr/lib/portage/bin/clean_locks для обработки блокировок (возникающих чаще всего после отказа системы или отключения).

ebuild-locks

Использовать блокировку, чтобы фазы сборки без sandbox никогда не выполнялись корректно. См. также *parallel-install*.

fakeroot

Включить fakeroot для установки и фаз установки пакетов, когда команду [ebuild](#)(1) запускает обычный пользователь.

fail-clean

Удалить временные файлы после неудачной сборки. Это особенно полезно, если ваша **PORTAGE_TMPDIR** на tmpfs. Если эта функция включена, вероятно, имеет смысл включить и **PORT_LOGDIR**, чтобы сохранялись журналы сборки. Команда [ebuild](#)(1) и функция *noclean* автоматически отключают *fail-clean*.

getbinpkg

Принудительно предпринимать попытку получить файлы от **_PORTAGE_BINHOST_**. Подробнее об этом см. [make.conf](#)(5). p((((*. *installsources* p(((((((. Устанавливает исходные коды в /usr/src/debug/\${CATEGORY}/\${PF} (см. также **splitdebug**). Эта функция работает только при условии, что установлен debugedit, а **CFLAGS** включает вывод отладочной информации (например, с флагом -ggdb).

keeptemp

Не удалять каталог \${T} по завершении процесса установки.

keepwork

Не удалять каталог \${WORKDIR} после установки пакетов. В этом случае \${WORKDIR} может быть использован повторно, поскольку данная функция фактически отключает очистку, которая обычно выполняется перед каждой сборкой. Из-за отсутствия надлежащей очистки эта функция может препятствовать нормальной отработке команды **emerge**, и поэтому ее не следует включать надолго.

fixlafiles

Изменяет файлы с расширением .la, чтобы не включать другие .la-файлы и некоторые другие исправления (порядок флагов, дубли записей и др.)

force-mirror

Забирать файлы только с настроенных зеркал, игнорируя **SRC_URI**, за исключением тех случаев, когда в переменной **RESTRICT** [ebuild](#)(5) указано *mirror*.

lmirror

Загружать файлы, когда в **FEATURES** включена функция *mirror*, даже если *mirror* выставлено в переменной **RESTRICT** [ebuild](#)(5). **НЕ ИСПОЛЬЗУЙТЕ** функцию *lmirror* для клиентов, которые должны иметь приоритет над установленным значением **RESTRICT** при скачивании с локального зеркала; вместо этого используйте настройку зеркала как локального (local) в файле */etc/portage/mirrors*, как описано в руководстве [portage](#)(5).

metadata-transfer

Автоматически выполнять передачу метаданных при запуске `emerge --sync`. В версиях portage выше 2.1.5, эта функция отключена по умолчанию. Если передача метаданных отключена, будет использоваться кэш метаданных непосредственно из каталога \${PORTDIR}/metadata/cache/ (при наличии); еклассы в \${PORTDIR}/eclass/ могут быть изменены исключительно в результате операции `emerge --sync`, поскольку механизм проверки кэша не распознает модификации еклассов. Как правило, с этим сталкиваются только пользователи rsync-дерева, поскольку дерево cvs не содержит каталога metadata/cache/. Если вы работаете с деревом rsync и хотите изменить дополнительные классы, пользуйтесь **PORTDIR_OVERLAY** для корректной проверки кэша.

mirror

Обращаться по всем адресам **SRC_URI** независимо от настроек **USE**, если только переменная **RESTRICT** не содержит *mirror* - в таком случае вообще не забирать файлы.

multilib-strict

Для многих Makefiles подразумевается, что их библиотеки находятся в каталоге /usr/lib либо \$(prefix)/lib. Если /usr/lib - не ссылка, указывающая на /usr/lib64, это может стать причиной серьезных проблем. В этом случае для обнаружения сбоящих пакетов portage обращается к функции *multilib-strict*. Если она используется, emerge всегда будет помещать 64-битные библиотеки в каталог (/usr)/lib64.

news

Включить поддержку новостей GLEP 42. См. <http://www.gentoo.org/proj/en/glep/glep-0042.html>.

noauto

При использовании **ebuild**(1) просто запускает запрошенную функцию.

noclean

По завершении установки не удалять исходные коды и временные файлы.

nodoc

Не устанавливать файлы документации (/usr/share/doc).

noinfo

Не устанавливать info-страницы.

noman

Не устанавливать man-руководства.

nostrip

Не удалять отладочную информацию из бинарных файлов, устанавливаемых в рабочей файловой системе.

notitles

Отключает обновление строки заголовка в xterm (содержит информацию о состоянии).

parallel-fetch

Загрузка файлов в фоновом режиме при компиляции. Для наблюдения за ходом этого процесса запустите `tail -f /var/log/emerge-fetch.log` в другом терминале.

parallel-install

При установке пакетов использует более тонкую блокировку, чтобы увеличить количество выполняемых потоков. Для дополнительного распараллеливания используйте *ebuild-locks*.

parse-eapi-ebuild-head

Анализировать **EAPI** начиная от заголовка ебилда(первые 30 строк). Это экспериментальная опция; в нормальных условиях ее не следует включать.

prelink-checksums

Если установлена утилита **prelink**(8), она будет использована для отмены предварительного связывания файлов перед вычислением контрольных сумм для установки и удаления. Эта функция полезна только в том случае, если установлен **prelink**(8) и по какой-либо причине требуются (несмотря на предварительное связывание) точные контрольные суммы - например, для проверки целостности установленных файлов или при отключенной функции *unmerge-orphans*.

Заметьте, что при нормальной установке пакетов с компиляцией из исходного кода контрольные суммы вычисляются до связывания, поэтому здесь данная функция также не нужна. Отказ от предварительного связывания при установке требуется только при использовании таких инструментов как **quickpkg**(1), которые могут запустить установку уже связанных файлов.

preserve-libs

Сохранять библиотеки при изменении, в ходе обновления или отката, имен .so-файлов. Библиотеки будут сохранены только в том случае, если обнаружены использующие их пакеты.

protect-owned

Данная функция аналогична *collision-protect*, с той разницей, что если файлы не прописаны явно в устанавливаемом пакете, они могут быть перезаписаны. Это особенно удобно для систем, содержащих большое количество неиспользуемых файлов, оставшихся от устаревших версий portage, где функция *unmerge-orphans* еще не поддерживалась. Для выборочного отключения данной функции, аналогично *collision-protect*, может быть использована переменная `_COLLISION_IGNORE_`. Рекомендуется всегда оставлять включенной либо *protect-owned*, либо *collision-protect*, поскольку в противном случае конфликты между файлами пакетов могут привести к несвоевременной перезаписи или удалению файлов. Если включена опция *collision-protect*, она будет иметь приоритет над *protect-owned*.

python-trace

Выводить подробный отчет о выполнении операций python на stderr, если для команды включена опция `--debug`.

sandbox

Включить *sandbox* при запуске [**emerge**\(1\)](#) и [**ebuild**\(1\)](#).

sesandbox

Включить *sandbox* для SELinux. Не изменяйте вручную эту функцию **FEATURE**.

sperms

Расшифровывается как Smart Filesystem Permissions ("умное управление правами в файловой системе"). Перед установкой пакетов в рабочей файловой системе автоматически осуществляет поиск и установку прав доступа к файлам setuid и setgid. У setuid-файлов будут сняты права на чтение для группы и остальных пользователей, а для setgid-файлов - только для остальных пользователей. См. также ниже описание `_suidctl_`.

sign

При коммитах в cvs с использованием [**repoman**\(1\)](#) подписывать Manifest-файл ключом GPG. О переменной `_PORTAGE_GPG_KEY_` см. в [**make.conf**\(5\)](#).

skiprocheck

Пропустить проверку доступа на запись к **DISTDIR** при загрузке файлов. Это полезно, если используются переменные **FETCHCOMMAND** и **RESUMECOMMAND**, обеспечивая перенаправление запросов на загрузку серверу, который предоставляет **DISTDIR** как общий ресурс NFS только для чтения. **DISTDIR**, доступный только для чтения, несовместим с опцией *distlocks*, поэтому рекомендуется добавить "*-distlocks*" в переменную **FEATURES**, чтобы избежать появления предупреждений, вызванных этой несовместимостью.

split-elog

Сохранять журналы, создаваемые **PORTAGE_ELOG_SYSTEM="save"** в подкаталогах категорий каталога **PORT_LOGDIR/elog**, а не непосредственно в **PORT_LOGDIR/elog**.

split-log

Хранить журналы сборки в подкаталогах категорий каталога **PORT_LOGDIR/build**, а не непосредственно в **PORT_LOGDIR**.

splitdebug

Прежде чем отладочная информация будет удалена из файлов ELF etdyn и etexec, она сохраняется, чтобы в дальнейшем при необходимости к ней мог был обратиться отладчик. Эту функцию можно отключить опцией **nostrip**. Для установки исходных кодов см. **installsources**.

strict

portage будет категорично реагировать на ситуации, которые могут представлять опасность (например, отсутствие или ошибки digest-файлов ебилдов).

stricter

portage будет категорично реагировать на ситуации, которые могут создавать конфликт настроек системы безопасности (например, text relocation, исполняемый стек). О переменных _QA_STRICT_*_ variables см. в руководстве по [make.conf\(5\)](#).

suidctl

Прежде чем устанавливать пакеты в рабочей файловой системе, автоматически удаляет данные о setuid из любого файла, не фигурирующего в */etc/portage/suidctl.conf*.

test

Запускать для пакетов специфичные тесты при каждой сборке, дабы убедиться, что пакет был скомпилирован без ошибок. См. *test* в [ebuild\(1\)](#) и *_src_test()* в [ebuild\(5\)](#). Эта функция подразумевает, что включен **USE-флаг test**.

test-fail-continue

Если переменная **FEATURES** содержит значение "test", а фаза проверки ебилда завершается с ошибкой, продолжать выполнение оставшихся фаз, как если бы ошибки не было. Обратите внимание, что фазу проверки можно отключить выборочно для отдельных пакетов, замаскировав **USE-флаг** в файле *package.use.mask* (см. [portage\(5\)](#)).

unmerge-logs

Сохранять журналы успешных удалений. Это имеет смысл только в том случае, если установлена переменная **PORT_LOGDIR**.

unmerge-orphans

Если файл не требуется другим пакетом в том же слоте и не защищен согласно **_CONFIG_PROTECT**, удалить его даже в том случае, если время последнего изменения или контрольная сумма отличаются от соответственных параметров изначально установленного файла.

unknown-features-filter

Отфильтровать все неизвестные значения в переменной **FEATURES**.

unknown-features-warn

Предупреждать, если переменная **FEATURES** содержит одно или несколько неизвестных значений.

userfetch

При запуске portage от имени пользователя root на время загрузки исходных кодов пакетов передавать права *portage:portage*.

userpriv

Разрешить portage отказаться от прав root и компилировать пакеты как *portage:portage*, без *sandbox* (если только одновременно не включена функция *usersandbox*).

usersandbox

Включить *sandbox* в фазу компиляции при работе без прав root (*userpriv*).

usersync

Предоставить владельцу **PORTDIR** права для выполнения **emerge --sync**.

webrsync-gpg

Включить проверку GPG при использовании *emerge-webrsync*.

FETCHCOMMAND

Эта переменная содержит команду, используемую для получения исходных кодов пакетов из Интернета. Она должна содержать полный путь доступа к исполняемому файлу, а также подстановочные *\\${DISTDIR}*, *\\${FILE}* и *\\${URI}*. Запись команды должна обеспечивать помещение загруженного файла по адресу *\\${DISTDIR}\\${FILE}*. См. также **RESUMECOMMAND**.

FFLAGS, FCFLAGS

Используйте эти переменные, чтобы выставить по вашему желанию настройки оптимизации/инструкции процессору для приложений, собираемых компилятором FORTRAN. Переменная FFLAGS обычно передается компилятору FORTRAN 77, тогда как переменная FCFLAGS используется на более современных системах сборки с любым компилятором FORTRAN.

Почитать об этом подробнее можно в разделе *Invoking GCC* ("Вызов GCC") тан-руководства по gcc: <http://gcc.gnu.org/onlinedocs/>

GENTOO_MIRRORS = [список URI]

Вставьте здесь, через пробел, список локальных зеркал. Они будут использоваться для загрузки файлов, имея приоритет перед перечисленными в *сценариях ebuild*. Полезной может оказаться утилита mirrorselect. Записи в данной переменной, не имеющие протокола и начинающиеся с разделителя пути '/', могут быть использованы для определения подмонтированных зеркал файловой системы.

http_proxy, ftp_proxy = [протокол://хост:порт]

Эти переменные используются в том случае, исходные коды должны доставляться из Сети с помощью **wget(1)**. Это необходимо лишь в том случае, если для выхода в Интернет вы используете прокси-сервер.

INSTALL_MASK = [файлы, через пробел]

Используйте эту переменную для выборочного запрета копирования файлов в ваше дерево файловой системы. Заметьте, что обрабатываются при этом будут только сами файлы, но не ссылки на них. Данная возможность полезна, в частности, в том случае, если вам необходимо отфильтровать файлы HACKING.gz и TODO.gz. Переменная **INSTALL_MASK** обрабатывается непосредственно перед установкой пакета. Поддерживается также переменная **PKG_INSTALL_MASK**, которая работает аналогично **INSTALL_MASK**, но обрабатывается непосредственно перед созданием бинарного пакета.

LDFLAGS

Список флагов, которые будут переданы компилятору при вызове компоновщика. Флаги компоновщика см. в справке по **ld(1)**; имейте в виду, что эти флаги будут переданы непосредственно компилятору. Поэтому следует использовать опцию '-Wl', чтобы избежать использования флагов, которые распознает только компоновщик (см. **gcc(1)**).

внимание

Установка этой и других переменных *FLAGS может вызвать отказ выполнения или компиляции. Если вы размещаете заявку на багтерекере, используя нестандартные значения данных опций, она может быть закрыта как недопустимая (INVALID).

MAKEOPTS

Используйте эту переменную для параллельной сборки. Например, если у вас двухъядерный процессор, вы можете установить ее значение на "-j2" или "-j3" для оптимизации сборки нескольких пакетов. Рекомендуемые значения находятся в диапазоне от **число процессоров+1** до **2*число процессоров+1**. Во избежание чрезмерной многопоточности следует использовать опцию **--load-average**. Подробнее см. **make(1)**. См. также сведения об аналогичных опциях **--jobs** и **--load-average** в справочном руководстве по **emerge(1)**. *NOCOLOR* = _["true" | "false"]_ p((((. Определяет, отключен ли по умолчанию цветной вывод. По умолчанию - "ложно" (false). *PKGDIR* =_[путь]_ p((((. Определяет каталог, в который будут помещены созданные бинарные пакеты .tbz2 binary, если включена опция ***emerge(1)** *--buildpkg*. По умолчанию каждый пакет помещается в подкаталог, соответствующий его категории. Тем не менее для обратной совместимости с расположением, используемым более ранними версиями portage, если каталог _\${PKGDIR}/All_ существует, все пакеты будут храниться в нем, а символические ссылки на пакеты будут созданы в подкаталогах категорий. Учтите, что подкаталоги /usr/portage могут оказаться небезопасны для хранения данных. Подробнее см. в справочном описании *PORTDIR*. По умолчанию это каталог /usr/portage/packages. *PORT_LOGDIR* p((((. Эта переменная определяет каталог, в котором хранятся журналы ебилдов; если переменная не определена, журналы создаваться не будут. Каждый журнал представляет собой файл формата \${CATEGORY}:\${PF}:YYYYMMDD-HHMMSS.log, лежащий в указанном каталоге. Если каталог не существует, он будет создан автоматически и получит права группы. Если каталог уже существует, права доступа к нему не будут изменены. *PORT_LOGDIR_CLEAN* p((((. Эта переменная должна содержать команду, с помощью которой portage

очистит PORT_LOGDIR. Стока команды должна содержать подстановочный \\$ {PORT_LOGDIR}, который будет заменен значением данной переменной. Переменная будет рабочей только в том случае, если в *FEATURES* назначена функция *clean-logs*. *PORTAGE_BINHOST* = _[список URI, через пробел]_ p(((. Это список хостов, с которых portage будет забирать прекомпилированные, бинарные пакеты. Каждый элемент списка должен представлять собой полной адрес каталога для работы с tbz2-архивами в вашей системе. Используется только в том случае, если *emerge* передаются опции, обеспечивающие выборку бинарных пакетов. Более подробная информация содержится в руководстве по *emerge*(1). В версиях portage ниже 2.1.6 эта переменная указывает на каталог All на хосте, который создает бинарные пакеты, а не на корневой каталог *PKGDIR*. Начиная с portage-2.1.6 она должна указывать на каталог, содержащий индексный файл Packages. Если файл \${PORTAGE_BINHOST}/Packages не существует, portage попытается использовать более ранний протокол. *PORTAGE_BINHOST_HEADER_URI* = _"ftp://login:pass@grp.mirror.site/pub/grp/i686/athlon-xp/"_ p(((. Эта переменная имеет смысл только в системе, которая будет выступать в качестве binhost и собирать пакеты для клиентов. Она определяет поле заголовка URI для индексного файла пакетов, который расположен по адресу \${PKGDIR}/Packages. Клиенты с правильным настройками PORTAGE_BINHOST смогут извлечь индекс и будут использовать поле заголовка URI для доставки бинарных пакетов. Если поле заголовка URI не определено, клиент будет использовать \${PORTAGE_BINHOST}, настроенный как основной URI.

PORTAGE_BINPKG_TAR_OPTS

Эта переменная содержит опции, которые будут передаваться команде tar для создания бинарных пакетов.

PORTAGE_BUNZIP2_COMMAND = [строка команд bunzip2]

Эта переменная должна содержать команду, с помощью которой portage будет производить извлечение файлов из bzip2-архива.

PORTAGE_BZIP2_COMMAND = [строка команд bzip2]

Эта переменная должна содержать команду, с помощью которой portage будет производить сжатие в архив bzip2. **PORTAGE_BZIP2_COMMAND** будет вызываться и для извлечения архива (для этого необходимо использовать опцию -d), при условии, что не установлена переменная **PORTAGE_BUNZIP2_COMMAND**.

PORTAGE_COMPRESS = "bzip2"

Эта переменная содержит команду, используемую для сжатия файлов документации во время фазы установки.

PORTAGE_COMPRESS_FLAGS = "-9"

Эта переменная содержит флаги, передаваемые команде **PORTAGE_COMPRESS**.

PORTAGE_COMPRESS_EXCLUDE_SUFFIXES = "gif htm[l]?jp[e]?g pdf png"

Эта переменная содержит, через пробел, перечень шаблонов суффиксов, по которым происходит исключение файлов при вызове команды **PORTAGE_COMPRESS**. Поддерживаются регулярные выражения; поиск соответствий осуществляется только по фрагменту имени файла, следующего за последним символом точки.

PORTAGE_ELOG_CLASSES

PORTAGE_ELOG_SYSTEM

PORTAGE_ELOG_COMMAND

PORTAGE_ELOG_MAILURI

PORTAGE_ELOG_MAILFROM

PORTAGE_ELOG_MAILSUBJECT

Обратитесь к справке по elog в файле /usr/share/portage/config/make.conf.example.

PORTAGE_FETCH_CHECKSUM_TRY_MIRRORS = 5

Число зеркал, с которых будет предпринята попытка повторного скачивания, если у загруженного файла ошибочная контрольная сумма.

PORTAGE_FETCH_RESUME_MIN_SIZE = 350K

Минимальный размер существующего файла, к которому обращается **RESUMECOMMAND**. Файлы размером меньше заданного будут удалены, будет вызван **FETCHCOMMAND**, а файлы будут загружены вновь, с самого начала. Это позволяет легко избавляться от небольших мусорных файлов - например, от html-страниц с ошибкой 404. Значением данной переменной должно быть целое число байтов, с суффиксом K (килобайты), M (мегабайты) или G (гигабайты).

PORTAGE_GPG_DIR

Домашний каталог **gpg(1)**, используемый утилитой **repoman(1)**, если в **FEATURES** выставлена опция **sign**.

По умолчанию это `$HOME/.gnupg`.

PORTAGE_GPG_KEY

Ключ **gpg(1)**, используемый утилитой **repoman(1)** для подписи manifest-файлов, если в ***FEATURES*** выставлена опция **sign**.

PORTAGE_GPG_SIGNING_COMMAND

Команда, используемая утилитой **repoman(1)** для подписи manifest-файлов, если в ***FEATURES*** выставлена опция **sign**.

PORTAGE_IONICE_COMMAND = [строка команд ionice]

Эта переменная должна содержать команду, которую portage будет вызывать для регулировки IO-приоритета самого portage и его подпроцессов. Страна команд должна содержать подстановочный `\${PID}`, которому соответствует целое PID. Так, значение "ionice -c 3 -p \\${PID}" устанавливает приоритет ожидания ввода-вывода. Подробнее об ionice см. **ionice(1)**. Данная переменная не установлена по умолчанию.

PORTAGE_NICENESS = [число]

Значение этой переменной будет добавлено к текущему nice-уровню, на котором работает emerge. Иными словами, она не устанавливает nice-уровень, а повышает его. Для получения дополнительной информации о nice-уровнях и допустимых значениях, см. **nice(1)**.

PORTAGE_RO_DISTDIRS = [каталоги, через пробел]

Если тот или иной файл не существует в каталоге **DISTDIR**, система будет искать его в каталогах из этого списка. Поиск по списку осуществляется слева направо. Обратите внимание, что текущая реализация работает путем создания символической ссылки в каталоге **DISTDIR**, но в будущем это может измениться.

PORTAGE_RSYNC_INITIAL_TIMEOUT = целое\число_

Определяет время ожидания перед первоначальным соединением с rsync-сервером при выполнении команды **emerge --sync**.

По умолчанию - 15 секунд.

PORTAGE_RSYNC_EXTRA_OPTS = [rsync options string]

Дополнительные опции rsync, которые будут передаваться команде **emerge --sync**.

Значение по умолчанию отсутствует.

PORTAGE_RSYNC_OPTS = [строка опций rsync]

Определяет стандартные опции rsync, которые будут передаваться команде **emerge --sync**.

Не редактируйте значение этой переменной, если не уверены в своих действиях!

По умолчанию - "--recursive --links --safe-links --perms --times --compress --force --whole-file --delete --stats --timeout=180 --exclude='/distfiles' --exclude='/local' --exclude='/packages'"

PORTAGE_RSYNC_RETRIES = [число]

Определяет максимальное количество повторных попыток соединения при синхронизации. Если выставить значением переменной отрицательное число, попытки соединения будут повторяться до тех пор, пока не будут исчерпаны все возможные адреса.

Значение по умолчанию составляет -1.

PORTAGE_SYNC_STALE = [число]

Определяет число дней после последнего выполнения `emerge --sync`, по истечении которых будет выводиться предупреждение. Если выставить значением переменной 0, предупреждения будут отключены.

Значение по умолчанию - 30.

PORTAGE_TMPDIR = [путь]

Определяет расположение временных каталогов сборки.

По умолчанию это каталог /var/tmp.

PORTAGE_WORKDIR_MODE = "0700"

Эта переменная управляет правами доступа к каталогу *WORKDIR* (см. [ebuild\(5\)](#)).

PORTDIR = [путь]

Определяет расположение дерева портежей - репозитория, где хранится вся информация о профиле и все ебилды. Если вы меняете значение этой переменной, вы должны соответственно изменить и символьическую ссылку /etc/make.profile.

По умолчанию это каталог /usr/portage.

*****Внимание*****

Данные, хранящиеся в **PORTDIR**, могут быть затерты или удалены при выполнении команды emerge --sync. Значение по умолчанию переменной **PORTAGE_RSYNC_OPTS** гарантирует, что сохраняются стандартные значения **DISTDIR** и **PKGDIR**, но пользователи должны отдавать себе отчет в том, что другие подкаталоги **PORTDIR** могут оказаться небезопасны для хранения данных. Не следует помещать иные данные (например, оверлеи) в каталог **PORTDIR**. Portage будет обходить структуру каталогов и может добавить произвольные нерабочие категории как пакеты.

PORTDIR_OVERLAY = "[путь] [другой-путь] [и т.д.]"

Определяет каталоги, в которых могут храниться ебилды, созданные пользователем, дабы они не перезаписывались при запуске `emerge --sync`. Список каталогов, через пробел.

Значение по умолчанию отсутствует.

QA_STRICT_EXECSTACK = "set"

Заставляет portage игнорировать любые настройки переопределения **_QA_EXECSTACK_** из ебилдов. См. также [ebuild\(5\)](#).

QA_STRICT_WX_LOAD = "set"

Заставляет portage игнорировать любые настройки переопределения **_QA_WX_LOAD_** из ебилдов. См. также [ebuild\(5\)](#).

QA_STRICT_TEXTRELS = "set"

Заставляет portage игнорировать любые настройки переопределения **_QA_TEXTREL_** из ебилдов. См. также [ebuild\(5\)](#).

QA_STRICT_DT_HASH = "set"

Заставляет portage игнорировать любые настройки переопределения **_QA_DT_HASH_** из ебилдов. См. также [ebuild\(5\)](#).

QA_STRICT_PRESTRIPPED = "set"

Заставляет portage игнорировать любые настройки переопределения **_QA_PRESTRIPPED_** из ебилдов. См. также [ebuild\(5\)](#).

RESUMECOMMAND

Эта переменная содержит команду, которая используется для возобновления частично выполненной загрузки исходных кодов. Она должна быть определена в том же формате, что и **FETCHCOMMAND**, и включать все дополнительные опции, которые могут понадобиться, чтобы докачать частично загруженный файл, расположенный в каталоге `\${DISTDIR}\${FILE}`.

ROOT = [путь]

Используйте переменную **ROOT** для определения целевой корневой файловой системы, которая будет использоваться для установки пакетов или ебилдов. Все **RDEPEND** и **PDEPEND** будут установлены в каталог **ROOT**, в то время как **DEPEND** - в каталог `/`. Обычно этот параметр устанавливается в окружении, а не непосредственно в файле `/etc/make.conf`; это удобно при создании нового образа сборки. Удостоверьтесь, что вы используете абсолютный путь.

Значение по умолчанию - корневой каталог, `/`.

RPMDIR = [путь]

Определяет каталог, в который будут помещаться создаваемые RPM-пакеты. По умолчанию это `\${PORTDIR}/rpm`.

SYNC = [RSYNC]

Эта переменная определяет предпочтаемое зеркало rsync. Указанный здесь rsync-сервер используется для синхронизации локального дерева портежей при запуске ``emerge --sync``.

Значение по умолчанию - `rsync://rsync.gentoo.org/gentoo-portage`

USE = [элементы USE, через пробел]

Эта переменная содержит опции, которые управляют сборкой ряда пакетов. Более подробную информацию см. в справке по [**ebuild**\(5\)](#). Список возможных значений USE имеется в файле `/usr/portage/profiles/use.desc`.

USE_EXPAND = [имена переменных, через пробел]

Любая переменная из этого списка будет использована для наращивания USE путем выставления нового флага для каждого элемента значения данной переменной; таким образом, результатом назначений `USE_EXPAND="FOO"` и `FOO="bar bla"` будет `USE="foo_bar foo_bla"`.

USE_EXPAND_HIDDEN = [имена переменных, через пробел]

Имена переменных **USE_EXPAND**, которые не должны отображаться в подробном выводе команды [**emerge**\(1\)](#).

USE_ORDER = "env:pkg:conf:defaults:pkginternal:repo:env.d"

Определяет порядок слоев в наращиваемом стеке переменной USE. Приоритет уменьшается слева направо: таким образом, значение env выше pkg, pkg выше conf, и так далее.

внимание

Если вы не разработчик и не уверены в том, что делаете, не изменяйте эту величину. Если вы это сделаете и возникнут неполадки, мы вам не поможем.

env

USE из текущих переменных окружения (USE и перечисленные в USE_EXPAND)

pkg

USE для определенных пакетов из `/etc/portage/package.use` (см. [**portage**\(5\)](#))

conf

USE из make.conf

defaults

USE из make.defaults и package.use профиля (см. [**portage**\(5\)](#))

pkginternal

USE из стандартных IUSE-настроек [**ebuild**\(5\)](#)

repo

USE из make.defaults и package.use репозитория (см. [portage\(5\)](#))

env.d

USE из переменных окружения, например, LINGUAS, определяемые файлами в каталоге /etc/env.d/

БАГТРЕКЕР

Об обнаруженных ошибках сообщайте на <http://bugs.gentoo.org/>

АВТОРЫ

- Daniel Robbins [\[drobbins@gentoo.org\]\(mailto:drobbins@gentoo.org\)\](mailto:drobbins@gentoo.org)
- Nicholas Jones [\[carpaski@gentoo.org\]\(mailto:carpaski@gentoo.org\)\](mailto:carpaski@gentoo.org)
- Mike Frysinger [\[vapier@gentoo.org\]\(mailto:vapier@gentoo.org\)\](mailto:vapier@gentoo.org)
- Saleem Abdulrasool [\[compnerd@gentoo.org\]\(mailto:compnerd@gentoo.org\)\](mailto:compnerd@gentoo.org)

ФАЙЛЫ

/etc/make.conf и /etc/portage/make.conf

Содержат переменные сборки, имеющие приоритет над переменными в make.defaults.

/usr/share/portage/config/make.globals

Содержит стандартные переменные сборки; редактируйте не этот файл, а /etc/make.conf.

/etc/portage/color.map

Содержит переменные для пользовательских настроек цвета.

/usr/portage/profiles/use.desc

Содержит полный список глобальных USE-переменных.

/usr/portage/profiles/use.local.desc

Содержит полный список локальных USE-переменных.

СМ. ТАКЖЕ

[emerge\(1\)](#), [portage\(5\)](#), [ebuild\(1\)](#), [ebuild\(5\)](#)

Сценарий /usr/sbin/ebuild.sh.

Вспомогательные приложения в каталоге /usr/lib/portage/bin.

ПЕРЕВОД

- Елена Гаврилова [\[e.vl.gavrilova@yandex.ru\]\(mailto:e.vl.gavrilova@yandex.ru\)\](mailto:e.vl.gavrilova@yandex.ru)

Август 2011

PORTRAGE

НАЗВАНИЕ

portage - главный инструмент Gentoo

ОПИСАНИЕ

В настоящее время portage использует большое количество конфигурационных файлов; со многими из них незнакомы пользователи и, во многих случаях, даже разработчики. В данном руководстве мы попытались собрать воедино все сведения о portage, которые позволили бы максимально эффективно использовать этот инструмент. Заметьте, что здесь мы описываем только те возможности, которые не имеют собственной справочной страницы.

Все файлы в каталоге make.profile могут быть тонко сконфигурированы при использовании каскадных профилей с помощью родительских профилей. Подробнее см. здесь:

<http://www.gentoo.org/proj/en/releng/docs/cascading-profiles.xml>

Примечание:

Данное руководство не содержит информации о том, как установить ту или иную программу; если вам нужна именно она, обратитесь к справке по [emerge\(1\)](#).

ПАРАМЕТРЫ

/etc/

make.conf (5)

/etc/make.profile/ или /etc/portage/make.profile/

значения для отдельных адресов помещаются в /etc/portage/profile/

deprecated

eapi

make.defaults

packages

packages.build

package.accept_keywords

package.keywords

package.mask

package.provided

package.unmask

package.use

package.use.force

package.use.mask

parent

profile.bashrc

use.force

use.mask

virtuals

/etc/portage/

bashrc

categories

color.map

license_groups

make.conf

mirrors

modules

package.accept_keywords

package.env

package.keywords

package.license

package.mask

package.properties

package.unmask

package.use

repos.conf

/etc/portage/env/

файлы конфигурации интерпретатора, bashrc, для отдельных пакетов

/etc/portage/profile/

переопределение **/etc/make.profile/** для отдельных адресов

/usr/portage/metadata/

layout.conf

/usr/portage/profiles/

```
arch.list
categories
info_pkgs
info_vars
license_groups
make.defaults
package.mask
package.unmask
package.use
package.use.force
package.use.mask
profiles.desc
repo_name
thirdpartymirrors
use.desc
use.force
use.local.desc
use.mask
```

/usr/share/portage/config/

make.globals

/var/cacheedb/

различные файлы внутреннего кэша

/var/db/pkg/

база данных для отслеживания установленных пакетов

/var/lib/portage/

config

world

world_sets

ГЛОССАРИЙ

В данном руководстве вы можете столкнуться с терминами, которые вам незнакомы или специфичны для Portage. Обращайтесь к указанным ниже man-страницам для более развернутой информации.

Атом зависимости

Атом имеет синтаксис следующего вида: либо *категория/пакет*, либо _*оператор категория/пакет*-*версия*. В качестве суффикса атом может иметь обозначение слота.

Подробнее см. [ebuild\(5\)](#).

Синтаксис расширенного атома

Следующие расширения синтаксиса атома поддерживаются только в пользовательских файлах конфигурации и в аргументах таких консольных команд как [emerge\(1\)](#):

Ограничения репозитория

Атомы с такими ограничениями содержат в конце, через разделитель '::', имя репозитория. Каждое имя репозитория должно соответствовать значению записи **repo_name** одного из репозиториев, сконфигурированных с помощью переменной **PORTDIR** или **PORTDIR_OVERLAY** (см. [make.conf\(5\)](#)).

Примеры:

```
# пакет sed из репозитория 'gentoo'  
sys-apps/sed::gentoo  
# пакет kdelibs из репозитория 'kde-testing'  
kde-base/kdelibs::kde-testing  
# пакет empathy из репозитория 'gnome'  
net-im/empathy::gnome
```

Шаблоны с подстановочными знаками

Атомы, содержащие подстановочные знаки, имеют синтаксис вида категория/пакет, где подстановочный знак '*' заменяет произвольное число из ряда допустимых. Разрешается использование нескольких символов '*', но они не должны стоять рядом друг с другом.

Примеры:

```
# любой пакет из категории 'sys-apps'  
sys-apps/*  
# любой пакет с именем 'zlib' из любой категории  
*/zlib  
# любой пакет из категории, начинающейся с 'net-'  
net-*/*  
# любой пакет из любой категории  
*/  
# любой пакет из репозитория 'gentoo'  
*/*:gentoo
```

KEYWORD

Каждая архитектура имеет свое ключевое слово (KEYWORD).

Подробнее см. [ebuild\(5\)](#)

virtual

Атом зависимости, который принадлежит категории виртуальных пакетов. Используются в том случае, когда разрешить необходимую зависимость могут сразу несколько пакетов, но требуется лишь один.
Подробнее см. [ebuild\(5\)](#)

НЕКОТОРЫЕ ВАЖНЫЕ ФАЙЛЫ

/etc/

make.conf

Глобальные редактируемые настройки, передаваемые Portage. См. [make.conf\(5\)](#).

/etc/make.profile/ или /etc/portage/make.profile/

Как правило, это лишь символическая ссылка на правильный профиль, хранящийся в каталоге **/usr/portage/profiles/**. Поскольку этот файл является частью дерева портежей, его можно легко обновить или сгенерировать заново, запустив `emerge --sync`. Он определяет профиль (обычно речь идет о параметрах, специфичных для данной архитектуры). Если вы хотите использовать свой особый профиль, вам необходимо создать свой собственный каталог **/etc/make.profile/** и заполнить его. Однако,

если вы намерены ограничиться переопределением некоторых настроек, используйте путь **/etc/portage/profile/** (он поддерживает те же типы файлов, что и **/etc/make.profile/**, за исключением родительских записей). НЕ РЕДАКТИРУЙТЕ настройки в **/etc/make.profile/**, поскольку после очередного `emerge --sync` они будут ПОТЕРЯНЫ. Если существуют два пути - и **/etc/make.profile/**, и **/etc/portage/make.profile/**, будет предпочтен **/etc/make.profile/**.

Любые файлы в этом каталоге, каталоги других профилей или высокоуровневые profiles-каталоги, имена которых начинаются с "package." или "use.", могут быть не только файлом, но и каталогом. Если это каталог, все файлы в нем будут отсортированы по имени в прямом алфавитном порядке и выведены вместе, как если бы они представляли собой один файл. Обратите внимание, что такое поведение поддерживается только в версиях portage выше 2.1.6.7, причем на данный момент оно не включено в PMS.

Пример:

```
 ${PORTDIR}/profiles/package.mask/removals  
 ${PORTDIR}/profiles/package.mask/testing
```

deprecated

Если этот файл присутствует, значит, профиль помечен как устаревший, более не поддерживаемый разработчиками Gentoo. В первой строке файла должен быть указан профиль, на который рекомендуется перейти пользователям, а ниже, возможно - инструкция, как это сделать.

Пример:

```
default-linux/x86/2005.0  
# emerge -n '>=sys-apps/portage-2.0.51'  
# rm -f /etc/make.profile  
# ln -s /usr/portage/profiles/default-linux/alpha/2005.0 /etc/make.profile
```

eapi

В первой строке этого файла указывается версия **EAPI** для всех файлов каталога. Справка по [ebuild\(5\)](#) предоставляет более развернутую информацию о **EAPI** и связанных с ним функциях.

make.defaults

Стандартные настройки профиля для Portage. Общий формат описан в руководстве по [make.conf\(5\)](#). Файл **make.defaults** для вашего профиля определяет несколько специфических переменных:

ARCH

Архитектура (x86/ppc/hppa и т.д.).

USERLAND = *GNU*

Поддержка BSD/cygwin и т.д.

ELIBC = *glibc*

Поддержка uClibc/BSD libc и т.д.

PROFILE_ONLY_VARIABLES = ARCH

Предотвращает изменение пользователем критических переменных в файлах make.conf или окружении.

PROFILE_ARCH

Позволяет различать классы машин с совпадающим значением **ARCH**. Так, для всех компьютеров Sparc справедливо **ARCH=sparc**, но данная переменная должна иметь значение 'sparc32' или 'sparc64'.

STAGE1_USE

Специальные USE-флаги, которые могут понадобиться при самогенерации stage2 из stage1.

packages

Перечень пакетов, входящих в состав сета *system*.

Формат:

- комментарии начинаются с символа # (внутритекстовые комментарии не допускаются)
- на одной строке может быть только один атом зависимости
- пакеты, которые должны быть добавлены в сет system, начинаются с символа *
- атомы без символа * выводятся только если речь идет об устаревших, но по-прежнему используемых пакетах

Примечание: При использовании каскадных профилей вы можете при настройке удалять в дочерних профилях пакеты, добавленные родительским профилем - для этого атом необходимо указать с префиксом '-'.

Пример:

```
# Это комментарий!
# загрузить версию glibc ниже 2.3
*<sys-libs/glibc-2.3
# загрузить любую версию bash
*app-shells/bash
# загрузить версию readline ниже 4.2
*<sys-libs/readline-4.2
```

packages.build

Список пакетов (по пакету на строку), которые составляют архив stage1. Его изменение имеет смысл только для разработчиков stage.

package.provided

Список пакетов (по пакету на строку), для которых portage предполагает, что они имеют источник данных. Имеет смысл при портировании на системы, отличные от Linux. В целом этот список заменяет **emerge --inject**.

Например, если вы работаете с собственной копией ядра 2.6, вы можете указать portage, что пакет 'sys-kernel/development-sources-2.6.7' уже обслуживается со стороны, так что он должен его игнорировать.

Portage не будет пытаться обновить пакет из данного списка, если только другой пакет явно не требует более новой его версии, чем указанная в списке. Зависимости, удовлетворяемые перечисленным в package.provided, могут вызвать удаление командой **emerge --depclean** установленных пакетов, удовлетворяющих соответствующие зависимости (подробнее см. в разделе **ДЕЙСТВИЯ** руководства [emerge\(1\)](#)).

В package.provided не должны помещаться виртуальные пакеты (virtual/*). В зависимости от типа такого пакета, вам может понадобиться добавить его в файл *virtuals* и/или добавить пакет, удовлетворяющий виртуальному, в package.provided.

Формат:

- комментарии начинаются с символа # (внутритекстовые комментарии не допускаются)
- на одной строке может быть только один атом зависимости
- операторы отношений не допускаются
- должна быть указана версия

Пример:

```
# вы занимаетесь разработкой ядра
sys-kernel/development-sources-2.6.7
```

```
# у вас установлена собственная копия QT  
x11-libs/qt-3.3.0

# у вас модульная среда X, а пакетам необходима монолитная  
x11-base/xorg-x11-6.8
```

package.use.force

Принудительное назначение USE-флагов для отдельных пакетов.

Примечание: При использовании каскадных профилей вы можете при настройке удалять в дочерних профилях USE-флаги, добавленные родительскими профилями - для этого флаг нужно указать с префиксом '-'.

Формат:

- комментарии начинаются с символа # (внутритекстовые комментарии не допускаются)
- по одному атому зависимости на строку, USE-флаги разделяются пробелом

Пример:

```
# принудительно установить документацию для GTK 2.x  
=x11-libs/gtk+-2* doc
# принудительно убрать поддержку mysql для QT
x11-libs/qt -mysql
```

package.use.mask

Маскировка USE-флагов для отдельных пакетов.

Примечание: При использовании каскадных профилей вы можете при настройке удалять в дочерних профилях USE-флаги, добавленные родительскими профилями - для этого флаг нужно указать с префиксом '-'.

Формат:

- комментарии начинаются с символа # (внутритекстовые комментарии не допускаются)
- по одному атому зависимости на строку, USE-флаги разделяются пробелом

Пример:

```
# замаскировать документацию для GTK 2.x  
=x11-libs/gtk+-2* doc
# размаскировать поддержку mysql для QT
x11-libs/qt -mysql
```

parent

Содержит путь доступа к родительскому профилю. Путь может быть как абсолютным, так и относительным. К местонахождению профиля путь будет относительным. Как правило, данный файл содержит указание на родительский каталог, '..'. Используется только в каскадных профильах.

profile.bashrc

При необходимости этот файл можно использовать для настройки особого окружения для билдов, отличного от стандартного для root. Синтаксис файла такой же, как для любого другого сценария bash.

use.force

В определенных обстоятельствах некоторые USE-флаги нет смысла отключать. Здесь перечисляются принудительно выставляемые флаги.

Примечание: При использовании каскадных профилей вы можете при настройке удалять в дочерних профилях USE-флаги, добавленные родительскими профилями - для этого флаг нужно указать с префиксом '-'.

Формат:

- комментарии начинаются с символа # (внутритекстовые комментарии не допускаются)
- по одному USE-флагу на строку

use.mask

Некоторые USE-флаги на отдельных архитектурах не имеют смысла (например, altivec на архитектуре, отличной от ppc, или mmx на архитектуре, отличной от x86) или не тестировались. Здесь перечисляются замаскированные флаги.

Примечание: При использовании каскадных профилей вы можете при настройке удалять в дочерних профилях USE-флаги, добавленные родительскими профилями - для этого флаг нужно указать с префиксом '-'.

Формат:

- комментарии начинаются с символа # (внутритекстовые комментарии не допускаются)
- по одному USE-флагу на строку

Пример:

```
# замаскировать doc
doc
# размаскировать mysql
-mysql
```

virtuals

Определяет, какие пакеты являются источником виртуальных по умолчанию. Например, если пакет должен отправлять почту, ему понадобится virtual/mta. Если нет пакета, предоставляющего virtual/mta (например, qmail, sendmail, postfix и т.д.), portage обращается к значению virtuals. В данном случае Gentoo стандартно использует net-mail/ssmtp (как определено в файле virtuals) - это пакет, который минимально необходим для отсылки электронных писем.

Формат:

- комментарии начинаются с символа # (внутритекстовые комментарии не допускаются)
- по одному виртуальному пакету и атому зависимости на строку

Пример:

```
# использовать net-mail/ssmtp как почтовый агент по умолчанию
virtual/mta          net-mail/ssmtp
# использовать app-dicts/aspell-en как словарь по умолчанию
virtual/aspell-dict  app-dicts/aspell-en
```

/etc/portage/

Любой элемент в этом каталоге, имя которого начинается с "package.", может быть не только файлом, но и каталогом. Если это каталог, все файлы, хранящиеся в нем, отсортированы по имени в прямом алфавитном порядке и сведены вместе, как если бы это был один файл.

Пример:

```
/etc/portage/package.accept_keywords/common  
/etc/portage/package.accept_keywords/e17  
/etc/portage/package.accept_keywords/kde
```

bashrc

При необходимости этот файл может быть использован для настройки особого окружения ebuildов, отличного от стандартного окружения root. По синтаксису файл представляет собой обычный сценарий bash.

В каталоге /etc/portage/env можно создавать дополнительные файлы настроек интерпретатора bashrc для отдельных пакетов.

categories

Простой список существующих категорий, которые могут быть использованы в /usr/portage, PORTDIR_OVERLAY и PKGDIR (см. [make.conf\(5\)](#)). Это позволяет создавать пользовательские категории.

Формат:

- по одной категории на строку

Пример:

```
app-hackers  
media-other
```

color.map

Содержит переменные для пользовательской настройки цветного вывода. См. [color.map\(5\)](#).

make.conf

Глобальные пользовательские настройки Portage. См. [make.conf\(5\)](#). Если этот файл существует, он имеет приоритет перед /etc/make.conf.

mirrors

Всякий раз, сталкиваясь с URI-ссылкой вида mirror://, portage обращается в поиске серверов сюда. Если mirrors не содержит перечня зеркал, portage обратится к глобальному файлу зеркал по адресу /usr/portage/profiles/thirdpartymirrors. Вы можете также указать особый тип зеркал, "local". Обращение к таким зеркалам происходит до считывания значения переменной GENTOO_MIRRORS; они используются даже в том случае, если для пакета установлены параметры RESTRICT="mirror" или RESTRICT="fetch".

Формат:

- комментарии начинаются с символа # (внутритекстовые комментарии не допускаются)
- сначала тип зеркала, затем список серверов

Пример:

```
# локальные частные зеркала моей компании  
local ftp://192.168.0.3/mirrors/gentoo http://192.168.0.4/distfiles
```

```
# если вы находитесь в Японии, предпочтительным для вас, скорее всего,  
будет японский сервер sourceforge  
sourceforge http://keihanna.dl.sourceforge.net/sourceforge  
  
# если вы находитесь в Тайване, предпочтительным для вас, скорее всего,  
будет местный сервер gnu  
gnu ftp://ftp.nctu.edu.tw/UNIX/gnu/
```

modules

Этот файл позволяет переопределять кэш метаданных. На практике единственной используемой здесь переменной, которую имеет смысл менять пользователю, является portdbapi.auxdbmodule.

Пример:

```
portdbapi.auxdbmodule = portage.cache.sqlite.database
```

После переопределения значения portdbapi.auxdbmodule может быть необходимо перенести или регенерировать кэш метаданных. Если вы используете дерево rsync и в [make.conf\(5\)](#) включено FEATURES="metadata-transfer", выполните `emerge --metadata`. Чтобы регенерировать метаданные для репозиториев из переменной **PORTDIR_OVERLAY** или дерева cvs, выполните `emerge --regen` (см. [emerge\(1\)](#)). Если вы используете, например, модуль sqlite и хотите, чтобы все метаданные сохранялись только в этом формате (удобно для запросов), включите FEATURES="metadata-transfer" в вашем [make.conf\(5\)](#).

package.accept_keywords и package.keywords

Позволяют назначать ACCEPT_KEYWORDS для отдельных пакетов. Это имеет смысл, если вам необходимо несколько нестабильных пакетов на стабильной системе или наоборот.

ACCEPT_KEYWORDS получает здесь аргументом один определенный пакет. Если в системе присутствуют и **package.accept_keywords**, и **package.keywords**, будут использованы оба, причем значения из **package.accept_keywords** будут иметь приоритет перед **package.keywords**. Файл **package.accept_keywords** призван заменить **package.keywords**, поскольку профили поддерживают другой формат **package.keywords**, который изменяет действующие значения KEYWORDS (а не ACCEPT_KEYWORDS).

Формат:

- комментарии начинаются с символа # (внутритекстовые комментарии не допускаются)
- в каждой строке - один атом зависимости и дополнительное ключевое слово
- если в строке не указано ключевых слов, подразумевается, что используется нестабильная ветка

Пример:

```
# всегда использовать нестабильные версии libgd  
media-libs/libgd ~x86  
# использовать только стабильные версии mplayer  
media-video/mplayer --x86  
# всегда использовать нестабильные версии netcat  
net-analyzer/netcat
```

Примечание: В дополнение к стандартным значениям переменной ACCEPT_KEYWORDS package.keywords поддерживает три особых маркера:

- * пакет отображается только в том случае, если он стабилен на любой архитектуре
- * пакет отображается только в том случае, если он тестируется на всех архитектурах

****** пакет отображается во всех случаях (ключевые слова игнорируются)

Дополнительное примечание: Если вы видите значение -* KEYWORD, это означает, что пакет не работает ни на одной системе, кроме перечисленных в KEYWORDS. Например, для пакета, существующего лишь как бинарный и собранного под архитектуру x86, имеем следующую запись:
games-fps/quake3-demo-1.11.ebuild:KEYWORDS="-* x86"

Если вы всё же хотите, чтобы этот пакет был доступен, используете другое ключевое слово в package.accept_keywords, например:

games-fps/quake3-demo x86

package.env

Переменные окружения для отдельных пакетов. Каждая запись соответствует файлу окружения; все эти файлы хранятся в каталоге /etc/portage/env/ и имеют тот же формат, что и [make.conf\(5\)](#).

Формат:

- комментарии начинаются с символа # (внутритекстовые комментарии не допускаются)
- в одной строке - один атом зависимости и соответствующий/-е ему файл(ы) окружения

Пример:

```
# использовать для пакета glibc переменные окружения из файла  
/etc/portage/env/glibc.conf  
sys-libs/glibc glibc.conf
```

package.license

Позволяет добавлять значения ACCEPT_LICENSE для отдельных пакетов.

Формат:

- комментарии начинаются с символа # (внутритекстовые комментарии не допускаются)
- в одной строке - один атом зависимости с дополнительными лицензиями или группами лицензий для него

package.mask

Список маскируемых атомов пакетов. Полезно в том случае, если не все версии требуемого пакета работают в вашей системе. Например, вам необходимы драйверы Nvidia drivers, но только версии до 1.0.4496. Нет ничего проще!

Формат:

- комментарии начинаются с символа # (внутритекстовые комментарии не допускаются)
- по одному атому зависимости на строку

Пример:

```
# замаскировать драйверы nvidia  
# версии 1.0.4496 и выше  
>=media-video/nvidia-kernel-1.0.4496  
>=media-video/nvidia-glx-1.0.4496
```

package.properties

Позволяет добавлять значения ACCEPT_PROPERTIES для отдельных пакетов.

Формат:

- комментарии начинаются с символа # (внутритекстовые комментарии не допускаются)
- в одной строке - один атом зависимости и дополнительные свойства для него

package.unmask

Аналогично описанному выше package.mask, но здесь, наоборот, перечисляются те пакеты, которые вы хотите размаскировать. Позволяет переопределить значения глобального файла package.mask (см. выше). Обратите внимание, что это не влияет на пакеты, которые были замаскированы посредством переменной KEYWORDS.

package.use

USE-флаги для отдельных пакетов. Удобно для отслеживания локальных USE-флагов, а также в случаях, когда флаги необходимо включать выборочно. Например, вы - разработчик GTK, поэтому вам необходима полная документация по данному проекту, а документация по QT не нужна. Элементарно, Ватсон.

Формат:

- комментарии начинаются с символа # (внутритекстовые комментарии не допускаются)
- в одной строке - один атом зависимости и его USE-флаги, через пробел

Пример:

```
# подключить документацию для GTK 2.x
=x11-libs/gtk+-2* doc
# отключить поддержку mysql для QT
x11-libs/qt -mysql
```

repos.conf

Содержит информацию о конфигурации *отдельных* репозиториев. Обратите внимание, что указанные здесь настройки конфигурации не относятся к таким утилитам как [repoman\(1\)](#) и [egencache\(1\)](#), поскольку выполняемые ими операции по определению **не зависят** от репозитория. **ВНИМАНИЕ:** в общем случае не рекомендуется использовать **repos.conf**, поскольку это повлечет за собой изменения в наследовании еклассов (прежде всего **переопределение еклассов**), которые, в свою очередь, в ряде случаев могут вызвать снижение производительности (см. ошибку #124041). При переопределении еклассов, дабы обойти ошибку #276264, вы должны убедиться, что ваше дерево портежей не содержит каталога metadata/cache/. Если же этот каталог существует, вам следует полностью его удалить и прописать значение переменной PORTAGE_RSYNC_EXTRA_OPTS="--exclude=/metadata/cache" в файле make.conf, чтобы исключить metadata/cache/ из операций *[emerge --sync](#).

Пример:

```
[DEFAULT]
# все репозитории будут наследовать еклассы из java-overlay и репозиториев
# java-experimental, причем еклассы из java-experimental
# будут иметь приоритет перед еклассами из java-overlay
eclass-overrides = java-overlay java-experimental

[gentoo]
# отключить переопределение еклассов для ебилдов из репозитория gentoo
eclass-overrides =
```

```

# при обработке данных metadata/layout.conf из других репозиториев,
вставлять
# 'gentoo' вместо алиасов-ссылок на репозитории 'foo' и 'bar',
# а алиас 'baz', содержащийся в файле layout.conf для gentoo, игнорировать
aliases = foo bar -baz

[kde-testing]
# переопределять настройки metadata/layout.conf основной ветки репозитория
kde-testing
masters = gentoo kde

[python]
# переопределять настройки metadata/layout.conf основной ветки репозитория
python,
# чтобы эти настройки не наследовались из ветки и чтобы
# репозиторий не тянул оттуда зависимости (пользователю необходимо
# убедиться, что все необходимые зависимости, такие как еклассы,
# удовлетворены)
masters =

```

/etc/portage/env/

В этом каталоге могут располагаться дополнительные файлы настройки интерпретатора, bashrc, для отдельных пакетов. Имейте в виду, что, если необходимы лишь настройки переменных окружения для этих пакетов, следует использовать не bashrc, а /etc/portage/package.env.

Функции set_unless_changed и unset_unless_changed functions можно использовать для изменения значений данных переменных, но только в том случае, если их значения не противоречат значениям, выставленным в файле make.conf. Это удобно, когда требуется временно переопределить переменные окружения при вызове emerge. Значения переменных, которых было назначено без использования функции set_unless_changed, всегда переопределяют значения, установленные при вызове emerge.

Синтаксис:

```

set_unless_changed ПЕРЕМЕННАЯ=ЗНАЧЕНИЕ
unset_unless_changed ЗНАЧЕНИЕ

```

Portage обрабатывает все файлы bashrc после /etc/portage/bashrc, в следующем порядке:

1. /etc/portage/env/\${CATEGORY}/\${PN}
2. /etc/portage/env/\${CATEGORY}/\${PN}: \${SLOT}
3. /etc/portage/env/\${CATEGORY}/\${P}
4. /etc/portage/env/\${CATEGORY}/\${PF}

/usr/portage/metadata/

layout.conf

Содержит схему репозиториев. Поддерживается атрибут masters, используемый для указания репозиториев, которые удовлетворяют зависимости еклассов и/или ебилдов. Каждому репозиторию должно соответствовать значение записи **repo_name** одного из репозиториев, определенного переменными **PORTEXPDIR** и **PORTEXPDIR_OVERLAY** (см. [make.conf\(5\)](#)). Репозитории справа от **masters** имеют большие приоритет, чем указанные слева. Поддерживается также атрибут aliases, поведение которого аналогично этому же атрибуту из файла repos.conf. Значения **layout.conf** можно переопределять для отдельных репозиториев в файле /etc/portage/repos.conf. Настройки, содержащиеся в **repos.conf**, имеют приоритет перед значениями **layout.conf**, однако такие утилиты как [repoman\(1\)](#) и [egecache\(1\)](#) полностью игнорируют **repos.conf**, поскольку выполняемые ими операции по определению **не зависят** от репозитория.

Пример:

```
# еклассы, предоставляемые java-overlay, имеют приоритет перед
одноименными
# екласами, предоставляемыми gentoo
masters = gentoo java-overlay
# указать, что данный репозиторий может заменять foo-overlay
aliases = foo-overlay
# не подписывать manifest-файлы в данном репозитории
sign-manifests = false
# использовать "легкие" manifest-файлы, содержащие только записи DIST
thin-manifests = true
# указать, что в данном репозитории каждому пакету должен соответствовать
manifest-файл, а если
# его нет или он некорректен, это будет рассматриваться как ошибка
use-manifests = strict
# указать, что для данного репозитория по умолчанию включена опция geroman
--echangelog=y
update-changelog = true
# указать, что данный репозиторий содержит файлы формата как md5-dict, так
и pms -
# их может генерировать egencache(1)
cache-formats = md5-dict pms
# указать, что данный репозиторий содержит профили, которые могут
использовать
# package.mask, package.provided, package.use, package.use.mask,
# package.use.force, use.mask и use.force как каталоги.
profile-formats = portage-1
```

/usr/portage/profiles/

Глобальные настройки Gentoo, контролируемые разработчиками. Переопределить их можно, используя файлы в каталоге /etc/portage/.

arch.list

Список всех доступных ключевых слов KEYWORDS. Модификаторы не учитываются.

Формат:

- по одному ключевому слову на строку

Пример:

```
x86
ppc
sparc
```

categories

Простой список доступных категорий, которые могут быть использованы в /usr/portage, PORTDIR_OVERLAY и PKGDIR (см. [make.conf\(5\)](#)).

Формат:

- по одной категории на строку

Пример:

```
app-admin
dev-lang
games-strategy
```

info_pkgs

Список всех пакетов, которые будут выведены по команде `emerge --info`.

info_vars

Список всех переменных, которые будут выведены по команде `emerge --info`.

license_groups

Содержит группы лицензий, которые могут быть определены переменной **ACCEPT_LICENSE** (см. [make.conf\(5\)](#)). Подробнее об этом см. в GLEP 23: <http://www.gentoo.org/proj/en/glep/glep-0023.html>.

Формат:

- комментарии начинаются с символа # (внутритекстовые комментарии не допускаются)
- в одной строке - одна группа с перечнем лицензий и вложенных групп
- вложенные группы должны иметь префикс @

Пример:

```
# Группа лицензий FSF-APPROVED полностью включает в себя группу GPL-COMPATIBLE и не только.  
FSF-APPROVED @GPL-COMPATIBLE Apache-1.1 BSD-4 MPL-1.0 MPL-1.1  
# Группа лицензий GPL-COMPATIBLE включает все лицензии, совместимые с GNU  
GPL.  
GPL-COMPATIBLE Apache-2.0 BSD BSD-2 GPL-2 GPL-3 LGPL-2.1 LGPL-3 X11 ZLIB
```

package.accept_keywords

Значения ACCEPT_KEYWORDS, используемые профилями, для отдельных пакетов. Формат и поведение этого файла аналогичны /etc/portage/package.accept_keywords, включая возможность включения в список атомов без ключевых слов, тем самым разрешая использовать нестабильные соответствия все стабильных веток, перечисленных в ACCEPT_KEYWORDS.

package.keywords

Значения KEYWORDS для отдельных профилей. Полезно в случаях, когда ключевое слово для того или иного пакета должно меняться в зависимости от выбранного профиля.

Формат:

- комментарии начинаются с символа # (внутритекстовые комментарии не допускаются)
- в одной строке - один атом с зависимостью с дополнительными ключевыми словами для него

Пример:

```
# добавить для libgd стабильную ветку  
media-libs/libgd x86  
# заменить для mplayer стабильную ветку нестабильной  
media-video/mplayer -x86 ~x86  
# удалить для netcat все ключевые слова  
net-analyzer/netcat -*
```

package.mask

Содержит перечень атомов зависимости для пакетов, которые не должны устанавливаться ни в одном профиле. Имеет смысл, например, если вам необходимо добавить последние бета-версии KDE, но не допустить, чтобы кто-либо из пользователей мог до них обновиться. Удобно и для быстрой маскировки отдельных версий, если были обнаружены проблемы с безопасностью. ВСЕГДА сопровождайте записи комментарием, в котором указывались бы ПРИЧИНА, по которой пакет замаскирован, и тот, КТО его замаскировал.

Формат:

- комментарии начинаются с символа # (внутритекстовые комментарии не допускаются)
- по одному атому зависимости на строку

Пример:

```
# замаскировать по соображениям безопасности
<sys-libs/zlib-1.1.4
# <caleb@gentoo.org> (10 Sep 2003)
# новые бета-версии kde
=kde-base/kde-3.2.0_beta1
=kde-base/kdeaccessibility-3.2.0_beta1
```

profiles.desc

Список всех текущих профилей, предназначенных для пользователей и разработчиков. Каждый указанный здесь профиль обрабатывается утилитой repoman.

Формат:

- комментарии начинаются с символа # (внутритекстовые комментарии не допускаются)
- в одной строке - один профиль, описанный в формате: _архитектура каталог состояния_, где:
 - архитектура должна фигурировать в arch.list
 - каталог должен фигурировать в profiles.desc
 - поддерживаются три состояния профиля: 'stable', 'dev' и 'exp'

Пример:

| | | |
|-----------|--------------------------|--------|
| alpha | default/linux/alpha/10.0 | stable |
| m68k | default/linux/m68k/10.0 | dev |
| x86 | default/linux/x86/10.0 | stable |
| x86-linux | prefix/linux/x86 | exp |

repo_name

Первая строка файла должна определять уникальное имя репозитория. Это имя может содержать любые символы из набора [A-Za-z0-9_-], причем дефис не может стоять в начале имени.

thirdpartymirrors

Управляет обработкой URI вида mirror://, представляющих собой доступные для загрузки зеркала. Предотвращает перегрузку какого-либо одного сервера.

Формат:

- комментарии начинаются с символа # (внутритекстовые комментарии не допускаются)
- в строке - тип сервера и перечень хостов

Пример:

```
sourceforge http://aleron.dl.sourceforge.net/sourceforge  
http://unc.dl.sourceforge.net/sourceforge
```

```
gentoo http://distro.ibiblio.org/pub/linux/distributions/gentoo/distfiles/  
ftp://ftp.gtlb.cc.gatech.edu/pub/gentoo/distfiles
```

```
kernel http://www.kernel.org/pub http://www.us.kernel.org/pub
```

use.desc

Здесь должны быть перечислены все глобальные USE-флаги с их описанием.

Формат:

- комментарии начинаются с символа # (внутритекстовые комментарии не допускаются)
- синтаксис строки: USE-флаг - описание

Пример:

```
3dfx - Adds support for 3dfx video cards  
acl - Adds support for Access Control Lists  
doc - Adds extra documentation
```

use.local.desc

Здесь перечисляются все локальные USE-флаги с описанием и указанием пакетов, для которых они установлены. Этот файл генерируется автоматически на основе файлов метаданных metadata.xml, включаемых во все пакеты. Подробнее см. в GLEP 56: <http://www.gentoo.org/proj/en/glep/glep-0056.html>.

Формат:

- комментарии начинаются с символа # (внутритекстовые комментарии не допускаются)
- синтаксис строки: пакет:USE-флаг - описание

Пример:

```
app-editors/nano:justify - Toggles the justify option  
dev-libs/DirectFB:fusion - Adds Multi Application support  
games-emulation/xmess:net - Adds network support
```

/usr/share/portage/config/

make.globals

Глобальные настройки по умолчанию для Portage. Они берутся непосредственно из пакета portage. Настройки, выставленные пользователем в **make.conf** или **package.env**, переопределяют их. Формат подробно описан в [make.conf\(5\)](#).

/var/cache/edb/

Этот каталог используется для хранения файлов внутреннего кэша portage. Имена и назначение этих файлов специально не документируются, дабы сэкономить ресурсы. Если вы не ведете разработку portage, вас, скорее всего, не касается описанное ниже.

Содержимое данного каталога может быть полностью удалено без какого-либо ущерба безопасности. Тем не менее убедительно рекомендуем не делать этого, поскольку повторное его генерирование может занять много времени.

/var/db/pkg/

Информация обо всех установленных пакетах хранится здесь. Portage определяет, установлен ли тот или иной пакет, путем обращения к этому каталогу.

Его структура в целом дублирует структуру дерева портежей: для каждой категории установленных пакетов предусмотрен подкаталог, а для каждой установленной версии пакета - вложенный в него каталог.

Каждый каталог пакета содержит различные файлы, описывающие установленные элементы пакета; здесь же хранится информация времени сборки (чтобы пакет можно было удалить, не обращаясь к дереву портежей).

Мы не описываем здесь точное содержимое и формат файлов, поскольку они могут в любой момент изменены. Впрочем, как правило, обычно каждой значимой переменной окружения (например, CFLAGS) соответствует один файл в данном каталоге. Обычно присутствует файл CONTENTS, содержащий адреса и хэши всех объектов, установленных пакетом в вашу систему.

/var/lib/portage/

config

Содержит хэши, позволяющие определить, изменились ли после установки файлы в каталогах, для которых используется система защиты конфигурационных файлов. Файлы, которые не были изменены, будут автоматически удалены из дерева.

world

Всякий раз, когда вы устанавливаете какой-либо пакет, его имя вносится в этот файл. При вызове `emerge world -up` из world-файла считывается список пакетов. Обратите внимание, что пакеты, установленные как зависимости, не вносятся в world. Например, если вы введете в терминале `emerge mod_wsgi`, а apache у вас еще не установлен, в world-файл будет внесен пакет "www-apache/mod_wsgi", но не "www-servers/apache". Подробнее см. [emerge\(1\)](#).

Формат:

- по одному атому зависимости на строку

Пример:

games-misc/fortune-mod-gentoo-dev

dev-libs/uclibc

app-cdr/cdemu

world_sets

Аналогичен world-файлу, но вместо атомов пакетов содержит сеты пакетов (всегда начинаются с символа @).

Пример:

@kde

БАГТРЕКЕР

Об обнаруженных ошибках сообщайте на <http://bugs.gentoo.org/>

АВТОРЫ

- Marius Mauch [\[genone@gentoo.org\]\(mailto:genone@gentoo.org\)\](mailto:genone@gentoo.org)
- Mike Frysinger [\[vapier@gentoo.org\]\(mailto:vapier@gentoo.org\)\](mailto:vapier@gentoo.org)
- Drake Wyrm [\[wyrm@haell.com\]\(mailto:wyrm@haell.com\)\](mailto:wyrm@haell.com)
- Arfrever Frehtes Taifersar Arahesis [\[arfrever@gentoo.org\]\(mailto:arfrever@gentoo.org\)\](mailto:arfrever@gentoo.org)

СМ. ТАКЖЕ

[emerge\(1\)](#), [ebuild\(1\)](#), [ebuild\(5\)](#), [make.conf\(5\)](#), [color.map\(5\)](#)

ПЕРЕВОД

- Елена Гаврилова [\[e.vl.gavrilova@yandex.ru\]\(mailto:e.vl.gavrilova@yandex.ru\)\](mailto:e.vl.gavrilova@yandex.ru)

Октябрь 2011

QUICKPKG

НАЗВАНИЕ

quicpkg - создание пакетов для дерева портей

СИНТАКСИС

quicpkg <список пакетов или сетов>

ОПИСАНИЕ

Утилита *quicpkg* может быть использована для быстрого создания пакета для portage на основе файлов, уже присутствующих в вашей системе. Вновь созданный пакет может затем быть развернут на любой другой системе. Для справки по синтаксису установки бинарных пакетов обратитесь к [emerge\(1\)](#). Преимущество этого метода состоит в том, что вам не нужно ждать, пока пакет будет распакован, сконфигурирован, скомпилирован и установлен, прежде чем вы сможете его использовать. Недостаток же заключается в том, что в состав пакета войдут файлы из вашей системы, даже если с момента их первой установки они были изменены.

После создания пакеты помещаются в **PKGDIR**. Данная переменная может быть определена в файле [make.conf\(5\)](#), а по умолчанию это /usr/portage/packages.

ОПЦИИ

<список пакетов или сетов>

Каждый пакет из списка может быть представлен двояко. Во-первых, вы можете указать полный путь к установленной записи в виртуальной базе данных: /var/db/pkg/<КАТЕГОРИЯ>/<ПАКЕТ-ВЕРСИЯ>. Во-вторых, допустимо использовать атом зависимости или сет пакетов portage. Атом или сет при этом будут выглядеть так, как если бы использовали их в качестве аргумента **emerge** при установке. Полное описание см. в [ebuild\(5\)](#).

ПРИМЕРЫ

```
quicpkg /var/db/pkg/dev-python/pyogg-1.1
quicpkg planeshift
quicpkg =apache-1.3.27-r1
quicpkg =net-www/apache-2*
quicpkg @system
```

БАГТРЕКЕР

Об обнаруженных ошибках сообщайте на <http://bugs.gentoo.org/>

АВТОРЫ

- Terry Chan (автор оригинальной версии)
- Mike Frysinger [\[vapier@gentoo.org\]\(mailto:vapier@gentoo.org\)\](mailto:vapier@gentoo.org) (переработанная версия)

ФАЙЛЫ

/etc/make.conf

Здесь определяется переменная **PKGDIR**.

СМ. ТАКЖЕ

[**ebuild**\(5\)](#), [**make.conf**\(5\)](#)

ПЕРЕВОД

- Елена Гаврилова [\[e.vl.gavrilova@yandex.ru\]\(mailto:e.vl.gavrilova@yandex.ru\)](mailto:e.vl.gavrilova@yandex.ru)

Март 2010

REPOMAN

НАЗВАНИЕ

repoman - программа Gentoo, обеспечивающая минимальный уровень качества пакетов, добавляемых к дереву портежей

СИНТАКСИС

repoman [*опция*] [*режим*]

ОПИСАНИЕ

Качество прежде всего.

Утилита **repoman** предназначена для контроля качества репозитариев ебилдов.

Примечание: **repoman commit** работает только в локальных репозитариях cvs, git или subversion.

ОПЦИИ

-a, --ask

Запросить подтверждение перед коммитом.

--force

Принудительно сделать коммит, независимо от проблем QA. Эта опция удобна, если вы хотите пропустить QA-проверки, занимающие наибольшее время. Сообщение, прилагаемое к коммиту, наряду с обычной отметкой о версии portage будет содержать указание на то, что опция включена.

При использовании в режиме **manifest** опция **--force** вызывает замену существующих digest-файлов любыми файлами, расположенными в каталоге `${DISTDIR}`. Существующие дайджесты будут считаться корректными для файлов, которые в противном случае потребовалось бы перезагрузить для повторной генерации дайджестов. **ВНИМАНИЕ:** При замене существующих дайджестов пользователю необходимо убедиться, что файлы в каталоге `${DISTDIR}` определяются верно. Особенно осторожным следует быть, если имеются частично загруженные файлы.

-q, --quiet

Не выдавать лишней информации.

-p, --pretend

Не делать коммита, не вносить исправлений, а только перечислить планируемые действия.

-x, --xmlparse

Принудительно выполнить синтаксический анализ `metadata.xml`.

-v, --verbose

При проверке отображать названия пакетов.

--changelog=

В режиме коммита вызывает `echangelog`, если чейнджлог не изменился (или, если выбрано значение `force` - независимо от изменений чейнджлога). Эта опция может быть включена по умолчанию для заданного репозитария путем установки значения `update-changelog = true` в файле `metadata/layout.conf` (см. [portage\(5\)](#)).

--if-modified=

Проверять только те пакеты, которые имеют изменения, не внесенные в коммит.

-i, --ignore-arches

Игнорировать ошибки, специфичные для данной архитектуры (архитектура != хост).

--ignore-default-opt

Не использовать переменную окружения `_REPOMAN_DEFAULT_OPTS_`.

-I, --ignore-masked

Игнорировать замаскированные пакеты (не допускается в режиме `commit`).

-d, --include-dev

Включить профили для разработчиков в проверку зависимостей.

--unmatched-removal

Включить строгую проверку файлов `package.mask` и `package.unmask` на предмет несовпадающих атомов на удаление.

--without-mask

Вести себя так, как если бы в `package.mask` не существовало записей (не допускается в режиме `commit`)

-m, --commitmsg

Добавить сообщение с коммитом в командной строке.

-M, --commitmsgfile

Добавить сообщение с коммитом из указанного файла.

-V, --version

Отобразить версию.

-h, --help

Вывести эту справку.

РЕЖИМЫ

full

Дерево каталога сканируется на предмет ошибок QA (полный вывод).

help

Выводится справка.

scan

Дерево каталога сканируется на предмет ошибок QA (краткий вывод).

fix

Исправляются простые ошибки QA (дайджест, не соотнесенный с пакетом, отсутствующий дайджест).

manifest

Генерируется `manifest`-файл (при необходимости будут загружены файлы исходного кода). Если вы хотите заменить существующие дайджесты, см. опцию `--force`.

manifest-check

Осуществляется проверка, не содержат ли манифесты отсутствующих или некорректных дайджестов.

commit

Дерево каталога будет просканировано на предмет ошибок QA; если ошибок нет, будет сделан коммит через cvs.

Ошибки QA

CVS/Entries.IO_error

При попытке сделать коммит возникла ошибка ввода-вывода при доступе к файлу Entries.

DEPEND.bad

Видимые пользователем ебилды с некорректным значением DEPEND (среди *видимых* ебилдов)

DEPEND.badindev

Видимые пользователем ебилды с некорректным значением DEPEND (среди *видимых* ебилдов) из ветки разработки

DEPEND.badmasked

Замаскированные ебилды с некорректным значением DEPEND (среди *всех* ебилдов)

DEPEND.badmaskedindev

Замаскированные ебилды с некорректным значением DEPEND (среди *всех* ебилдов) из ветки разработки

DEPEND.badtilde

DEPEND использует оператор зависимости \sim dep с ненулевой строкой ревизии, что не имеет смысла (ревизия игнорируется)

DEPEND.syntax

Синтаксическая ошибка в DEPEND (обычно лишний или недостающий пробел/скобка)

DESCRIPTION.missing

Ебилды без переменной DESCRIPTION или с пустым значением этой переменной.

EAPI.definition

EAPI определяется после наследования (а должен - до).

EAPI.deprecated

Ебилды с функционалом, устаревшим в текущей версии EAPI.

EAPI.incompatible

Ебилды с функционалом, доступным только с EAPI другой версии.

EAPI.unsupported

Ебилды с неподдерживаемой версией EAPI (вы должны обновить portage)

HOME PAGE.missing

Ебилды без переменной HOME PAGE или с пустым значением этой переменной.

HOME PAGE.virtual

Виртуальные пакеты с непустым значением переменной HOME PAGE.

IUSE.invalid

Ебилд вызывает IUSE-переменную, которая не фигурирует в use.desc или его файле метаданных metadata.xml.

IUSE.missing

Ебилд использует условие USE, которое отсылает к флагу, отсутствующему в IUSE.

IUSE.undefined

Ебилд не определяет IUSE (тогда как правила синтаксиса требуют определять IUSE даже при пустом значении).

KEYWORDS.dropped

Ебилды, для которых, вероятно, установленное значение KEYWORDS было заменено другим ключевым словом архитектуры.

KEYWORDS.invalid

Ебилд содержит ключевое слово, которое не указано в profiles/arch.list или на которое не был найден доступный профиль.

KEYWORDS.missing

Ебилды без переменной KEYWORDS или с пустым значением этой переменной.

KEYWORDS.stable

Ебилды, которые были напрямую добавлены с ключевым словом стабильной архитектуры.

KEYWORDS.stupid

Ебилды, использующие значение KEYWORDS=-* вместо значения файла package.mask.

LICENSE.invalid

Ебилд имеет лицензию, которая не фигурирует в каталоге license/ портежей.

LICENSE.missing

Ебилды без переменной LICENSE или с пустым значением этой переменной.

LICENSE.syntax

Синтаксическая ошибка в LICENSE (обычно лишний или недостающий пробел/скобка).

LICENSE.virtual

Виртуальные пакеты с непустым значением переменной LICENSE.

LIVEVCS.stable

Ебилд представляет собой ебилд live-системы контроля (cvs, git, darcs, svn и т.д.) с ключевым словом стабильной архитектуры.

LIVEVCS.unmasked

Ебилд представляет собой ебилд live-системы контроля (cvs, git, darcs, svn и т.д.), но имеет ключевые слова и не замаскирован в глобальном package.mask.

PDEPEND.bad

Видимые пользователям ебилды с некорректным значением PDEPEND (среди *видимых* ебилдов).

PDEPEND.badindev

Видимые пользователям ебилды с некорректным значением PDEPEND (среди *видимых* ебилдов) в ветке разработки.

PDEPEND.badmasked

Замаскированные ебилды с некорректным значением PDEPEND (среди *всех* ебилдов).

PDEPEND.badmaskedindev

Замаскированные ебилды с некорректным значением PDEPEND (среди *всех* ебилдов) в ветке разработки.

PDEPEND.badtilde

PDEPEND использует оператор зависимости \sim dep с ненулевой строкой ревизии, что не имеет смысла (ревизия игнорируется).

PDEPEND.suspect

PDEPEND содержит пакет, который обычно принадлежит только DEPEND.

PDEPEND.syntax

Синтаксическая ошибка в PDEPEND (обычно лишний или недостающий пробел/скобка).

PROVIDE.syntax

Синтаксическая ошибка в PROVIDE (обычно лишний или недостающий пробел/скобка).

RDEPEND.bad

Видимые пользователям ебилды с некорректным значением RDEPEND (среди *видимых* ебилдов).

RDEPEND.badindev

Видимые пользователям ебилды с некорректным значением RDEPEND (среди *видимых* ебилдов) в ветке разработки.

RDEPEND.badmasked

Замаскированные ебилды с некорректным значением RDEPEND (среди *всех* ебилдов).

RDEPEND.badmaskedindev

Замаскированные ебилды с некорректным значением RDEPEND (среди *всех* ебилдов) в ветке разработки.

RDEPEND.badtilde

RDEPEND использует оператор зависимости \sim dep с ненулевой строкой ревизии, что не имеет смысла (ревизия игнорируется).

RDEPEND.implicit

Значение RDEPEND в ебilde не установлено; тем самым неявно назначается RDEPEND=\$DEPEND (до EAPI 4).

RDEPEND.suspect

RDEPEND содержит пакет, который обычно принадлежит только DEPEND.

RDEPEND.syntax

Синтаксическая ошибка в RDEPEND (обычно лишний или недостающий пробел/скобка).

PROPERTIES.syntax

Синтаксическая ошибка в PROPERTIES (обычно лишний или недостающий пробел/скобка).

RESTRICT.syntax

Синтаксическая ошибка в RESTRICT (обычно лишний или недостающий пробел/скобка).

SLOT.invalid

Ебилды с отсутствующим или ошибочным значением переменной SLOT.

SRC_URI.mirror

URI, фигурирующий в profiles/thirdpartymirrors, обнаружен среди значений SRC_URI.

changelog.ebuildadded

Был добавлен ебилд, но чейнджлог не изменился.

changelog.missing

Отсутствуют чейнджлоги.

changelog.notadded

Чейнджлоги существуют, но не было добавлены в cvs.

dependency.unknown

Ебилд имеет в зависимостях неизвестный пакет (это может не быть ошибкой, если речь идет о блокировке переименованного/удаленного пакета или об альтернативе, предоставляемой оверлеем).

digest.assumed

Существующий дайджест должен считаться корректным (только на уровне пакета)

digest.missing

Некоторые файлы, указанные в SRC_URI, не фигурируют в манифесте.

digest.unused

Некоторые файлы, указанные в Manifest-файле, не фигурируют в SRC_URI.

ebuild.allmasked

Для этого пакета замаскированы все ебилды (только на уровне пакета).

ebuild.badheader

У ебилда некорректный заголовок.

ebuild.invalidname

Файлы ебилда имеют названия, не поддающиеся синтаксическому анализу или содержащие синтаксические ошибки (или имеют расширения версии portage 2.1).

ebuild:majorsyn

Ебилд содержит серьезную синтаксическую ошибку, которая может привести к полной или частичной неработоспособности ебилда.

ebuild.minorsyn

Ебилд содержит мелкую синтаксическую ошибку, нарушающую требования, предъявляемые к программированию под gentoo.

ebuild.namenomatch

Ебилд содержит файлы, имена которых не совпадают с именами в родительском каталоге.

ebuild.nesteddie

Помещение 'die' в () вызывает ошибку, но не завершает ebuild-сценарий.

ebuild.nostable

Нет ебилдов, которые были бы маркованы как стабильные для вашей архитектуры.

ebuild.notadded

Ебилды существуют, но не были добавлены в cvs.

ebuild.output

Простая передача данных ебилду сопровождается выводом; это нарушает требования, предъявляемые к ебилдам.

ebuild.patches

Переменная PATCHES должна быть массивом bash, чтобы обеспечивалась корректная обработка пробелов.

ebuild.syntax

Ошибка генерации кэша ебилда, вызванная, вероятнее всего, синтаксической ошибкой в ебилде или ошибкой при проверке дайджеста.

eprefixify.defined

Ебилд использует `eprefixify`, но не наследует екласc префикса.

file.UTF8

Файл не в кодировке UTF8.

file.executable

Ебилды, дайджесты, файл метаданных `metadata.xml`, манифест, чейнджлог не требуют исполняемого файла.

file.name

Имя файла/каталога может включать только следующие символы: `a-zA-Z0-9._-+`:

file.size

Файлы в каталоге `files` должны быть размером менее 20k.

inherit.autotools

Ебилд наследует `autotools`, но не вызывает `eautomake`, `eautoconf` и `eautoreconf`.

inherit.deprecated

Ебилд наследует устаревший екласc.

java.eclassesnotused

Если `virtual/jdk` прописан в `DEPEND`, должен наследоваться екласc `java`. См. подробнее:

<http://www.gentoo.org/proj/en/java/java-devel.xml>.

manifest.bad

Manifest-файл имеет отсутствующие или некорректные дайджесты.

metadata.bad

Некорректные файлы `metadata.xml`.

metadata.missing

Отсутствуют файлы `metadata.xml`.

metadata.warning

Предупреждения в файлах `metadata.xml`.

portage.internal

Ебилд обращается к внутренней функции Portage.

upstream.workaround

Ебилд обходит ошибку апстрима; на ошибку следует разместить заявку на багтрекере bugs.gentoo.org.

usage.obsolete

Ебилд использует устаревшую конструкцию.

variable.invalidchar

Переменная содержит некорректный символ, не из набора ASCII.

variable.readonly

Переменная назначается только для чтения.

variable.usedwithhelpers

Ебилд использует переменные `D`, `ROOT`, `ED`, `EROOT`, `EPREFIX` с помощниками.

virtual.oldstyle

Ебилд предоставляет виртуальный пакет устаревшего типа (см. GLEP 37).

wxwidgets.eclassesnotused

Ебилд зависит от пакета x11-libs/wxGTK, не наследуя класс wxwidgets.eclass. См. подробнее в отчете об ошибке #305469.

БАГТРЕКЕР

Об обнаруженных ошибках сообщайте на <http://bugs.gentoo.org>.

АВТОРЫ

- Daniel Robbins [\[drobbins@gentoo.org\]\(mailto:drobbins@gentoo.org\)](mailto:drobbins@gentoo.org)\
- Saleem Abdulrasool [\[compnerd@gentoo.org\]\(mailto:compnerd@gentoo.org\)](mailto:compnerd@gentoo.org)\

СМ. ТАКЖЕ

[emerge\(1\)](#)

ПЕРЕВОД

- Елена Гаврилова [\[e.vl.gavrilova@yandex.ru\]\(mailto:e.vl.gavrilova@yandex.ru\)](mailto:e.vl.gavrilova@yandex.ru)\

Октябрь 2011

XPAK

НАЗВАНИЕ

xpak - формат данных XPAK, используемый в бинарных пакетах Portage

КРАТКОЕ ОПИСАНИЕ

К каждому бинарному пакету Gentoo прилагаются данные xpak, которые содержат различную информацию времени сборки - например, USE-флаги, с которыми пакет был скомпилирован, исходный ебилд, переменные окружения, значения переменных CFLAGS и CXXFLAGS и т.д.

ПРИМЕЧАНИЯ

Типы данных

Ниже описаны все рассматриваемые в данной документации случаи.

Целое число

Все отступы/длины являются беззнаковыми 32-разрядными целыми числами, с порядком следования байт от старшего к младшему.

Строка

Все строки в кодировке ASCII и не оканчиваются на NUL (кавычки - только для иллюстрации)

Значение

Текущие значения отдельных записей xpak хранятся как строки.

Вертикальная черта

Вертикальная черта '|' не является частью формата файла; она используется лишь для иллюстрации того, как значения отступа применяются к данным.

СИНТАКСИС

бинарный пакет (tbz2)

```
|<-отступ_xpak->|
<tar>|<    xpak      >|<отступ_xpak>"STOP"
```

xpak

"ХРАКРАСК"<строка_индекса><строка_данных><индекс><данные>"ХРАКSTOP"

индекс

|----- строка_индекса----->|
<индекс1><индекс2><индекс3><индекс4><...>|

индексN

|<- строка_имени->|
<строка_имени><имя><отступ_данных><строка_данных>

данные

|----- строка_данных----->|
<-отступ_данныхN-><-строка_данныхN->|
<данные><данные_N><данные>|

ПОДРОБНОЕ ОПИСАНИЕ

xpak

Если вы рассмотрите любой бинарный пакет Gentoo с помощью шестнадцатиричного редактора, вы обнаружите, что он содержит собственно архив файлов, а далее бинарный блоб - *xpak*, отступ, содержащий байты от начала *xpak'a* до конца файла - *_отступ_xpak и, наконец, строку "STOP"*.

|<отступ_xpak>|
<tar><---xpak---><отступ_xpak>"STOP"

В приведенном примере вы можете видеть архив *tar*, связанный с ним *_блоб_xpak*, отступ_xpak и, в конце, строку "STOP". Эти метаданные не рассматриваются как часть *xpak*, а скорее как часть бинарного пакета.

Если мы возьмем значение оступа и отчитаем соответствующее ему количество байтов назад от начала *_отступа_xpak*, то придем к блоку *xpak*, который начинается со строки "ХРАКРАСК".

Блок *xpak* состоит из строки "ХРАКРАСК", длины блока *индекса* (строка\индекса), длины блока *данных* (строка_данных), бинарный блоб строки_индекса, содержащий *индекс*, бинарный блоб строки_данных, содержащий *данные*, и строки "ХРАКSTOP" в конце:

|<строка_индекса><строка_данных>|
"ХРАКРАСК"<строка_индекса><строка_данных><-- индекс--><-- данные-->| "ХРАКSTOP"

Чтобы получить *индекс* и *данные*, мы отсекаем с конца строки\данных количество байт, соответствующее строке_индекса (блок индекса), а затем - байты, соответствующие следующей строке_данных (блок данных). Если мы всё сделали правильно, следующие байты будут представлять собой строку "ХРАКSTOP" в формате ASCII.

Все *данные* сведены в один большой блок; таким образом, чтобы их считать, необходимо знать фактическое положение каждого фрагмента информации в этом блоке. Эту информацию можно получить с помощью индексов, хранящихся в блоке индекса.

Блок \индекса_

Блок \индекса_ включает ряд индексов:

|----- строка_индексов----->|

```
|<индекс1><индекс2><индекс3><индекс4><индекс5><индекс6><индекс7>|
```

Блок *индекса* содержит всю необходимую нам информацию для блока *данных*. Он содержит ряд отдельных индексов, которые все вместе составляют строку\индексов_. Здесь нет разделения нулем или тому подобного.

Каждый из этих элементов соответствует фрагменту данных в блоке *данных*: строка имени этого блока (*строка\имени*), длина _*строки_имени* в байтах, отступ блока (*отступ\данныхN*) и длина блока (_*строка_данныхN*):

```
|<строка_имени>|
<строка_имени>|<- - имя - ->|<отступ_данныхN><строка_данныхN>
```

Блок\данных_

Блок\данных содержит ряд фрагментов данных, которые в сумме образуют строку_данных_:

```
|<----- строка_данных ----->|
|
<данные1><данные2><данные3><данные4><данные5><данные6><данные7><данные . . . >
|
```

Для выбора одного элемента данных нам понадобится *отступ\данных* и *строка_данных* из индекса. Опираясь на них, мы можем рассчитать количество байтов в строке_данных от начала блока_данных, а затем отнять от них байты ближайшей следующей строки_данных_. Тем самым мы получаем наш исходный блок данных:

```
|<---- отступ_данныхN---->|<- - строка_данныхN - ->|
|<данные1данные2данные3данные . . . >|<нужные_нам_данные>|
```

ПРИМЕРЫ

Предположим, что у нас есть xrapk, содержащий два фрагмента данных. Один из них именуется "file1" и содержит строку "ddDddDdd", а другой - "file2" и содержит строку "jjJjjJjj". Данные не содержат "STOP" или _*отступ_xrapk*_, поскольку данный xrapk не является частью бинарного пакета.

Вот вывод шестнадцатиричных данных (построчно):

| | |
|--|-----------------------|
| 00 58 50 41 4b 50 41 43 4b 00 00 00 20 00 00 00 10 | XPAKРАСК |
| 10 00 00 00 04 66 69 6c 31 00 00 00 00 00 00 00 08 | . . . файл1 |
| 20 00 00 00 04 66 69 6c 32 00 00 00 08 00 00 00 08 | . . . файл2 |
| 30 64 64 44 64 64 44 64 64 6a 6a 4a 6a 6a 4a 6a 6a | ddDddDddjjJJjjJJjj |
| 40 58 50 41 4b 53 54 4f 50 | XPAKSTOP |

Строка\индекса имеет значение 32, а строка_данных_ - 16 (поскольку данные содержат 16 байт: "ddDddDdd" и "jjJjjJjj").

```
|<---- "XPAKРАСК"---->| | 32 | 16 |
00 58 50 41 4b 50 41 43 4b 00 00 00 20 00 00 00 10
```

А вот первый элемент индекса, значение строки\имени_ которого составляет 4, за ней идет строка "файл1", далее - отступ данных1 со значением 0 и данные1 со значением 8 (поскольку данные1 содержат 8 байт: "ddDddDdd"):

```
| 4 |<- - "файл1" - ->| | отступ_данных1:0 | строка_данных1:8 |
10 00 00 00 04 66 69 6c 31 00 00 00 00 00 00 00 08
```

Теперь рассмотрим второй элемент индекса, со значением строки\индекса 4; за ней идет строка индекса "файл2", отступ\данных2 со значением 8 и данные2 со значением 8 (поскольку данные2 содержат 8 байт: "jjJjjJjj").

```
|      4      |<-- "файл2"->| | отступ_данных2:8 | строка_данных2:8 |
20 00 00 00 04 66 69 6c 32 00 00 00 08 00 00 00 08
```

```
|<-----"XPAKSTOP"----->|
40 58 50 41 4b 53 54 4f 50
```

АВТОРЫ

- Lars Hartmann [\[lars@chaotika.org\]\(mailto:lars@chaotika.org\)\](mailto:lars@chaotika.org)
- Mike Frysinger [\[vapier@gentoo.org\]\(mailto:vapier@gentoo.org\)\](mailto:vapier@gentoo.org)

СМ. ТАКЖЕ

[qtbz2](#)(1), [quickpkg](#)(1), [qxpak](#)(1)

ПЕРЕВОД

- Елена Гаврилова [\[e.vl.gavrilova@yandex.ru\]\(mailto:e.vl.gavrilova@yandex.ru\)\](mailto:e.vl.gavrilova@yandex.ru)

Октябрь 2011

Справка по OpenRC

OpenRC ([sys-apps/openrc](#)) управляет службами системы, запуском и выходом.

- [rc](#) - остановка и запуск служб на определенном уровне запуска
- [rc-service](#) - обнаружение и запуск OpenRC-служб в заданными аргументами
- [rc-status](#) - отображение информации об уровнях запуска
- [rc-update](#) - добавление/удаление служб на том или ином уровне запуска
- [runscript](#) - интерпретатор сценариев оболочки для служб
- [start-stop-daemon](#) - обеспечение запуска и остановки демонов

RC

НАЗВАНИЕ

rc - останавливает и запускает службы на определенном уровне запуска

СИНТАКСИС

rc [-o , -override] [уровень запуска]

ОПИСАНИЕ

rc останавливает все службы, не принадлежащие уровню запуска, а затем запускает службы данного уровня и уровней, добавленных с помощью **rc-update**, но еще не включенных. Если уровень запуска не указан, будет использован текущий.

Существует ряд особых уровней запуска, о которых вы должны знать:

sysinit

Подключает такие системные каталоги как /dev, /proc и, возможно, /sys для Linux-систем. Кроме того, монтирует /lib/rc/init.d как электронный диск с использованием файловой системы tmpfs, когда она доступна, при условии, что корневой каталог / не монтируется при запуске с параметром rw. Информация о состоянии запущенных **rc** служб располагается в каталоге /lib/rc/init.d. **sysinit** всегда инициируется при включении машины и не должен быть иницирован повторно.

boot

Как правило, на уровень запуска boot добавляются лишь те службы, которые связаны с монтированием файловых систем, первоначальной настройкой подключаемых устройств и входом в систему. При горячем подключении на уровень boot добавляются соответствующие службы. Все службы на уровнях запуска boot и sysinit автоматически прописываются на все остальные уровни, кроме перечисленных здесь.

single

Останавливает все службы, кроме находящихся на уровне запуска sysinit.

reboot

Переключает на уровень запуска shutdown, а затем перезагружает машину.

shutdown

Переключает на уровень запуска shutdown, а затем останавливает машину.

Вы не должны вызывать перечисленные уровни запуска самостоятельно. Вместо этого используйте init(8) и shutdown(8) - именно они отвечают за вызов этих особых уровней.

СМ. ТАКЖЕ

[rc-status\(8\)](#), [rc-update\(8\)](#), [init\(8\)](#), [shutdown\(8\)](#)

АВТОРЫ

- Roy Marples [\[roy@marples.name\]\(mailto:roy@marples.name\)\](mailto:roy@marples.name)

ПЕРЕВОД

- Елена Гаврилова [\[e.vl.gavrilova@yandex.ru\]\(mailto:e.vl.gavrilova@yandex.ru\)\](mailto:e.vl.gavrilova@yandex.ru)

RC-SERVICE

НАЗВАНИЕ

rc-service - обнаружение и запуск OpenRC-служб в заданными аргументами

СИНТАКСИС

rc-service [-i , -ifexists] служба команда [...] rc-service -e , -exists служба rc-service -l , -list rc-service -r , -resolve служба

ОПИСАНИЕ

Сценарии служб могут располагаться в разных местах в зависимости от системы. **rc-service** позволяет найти запрашиваемую службу и запустить ее с заданными аргументами.

Если задан аргумент **-i , -ifexists**, то **rc-service** возвращает 0 даже в том случае, если служба не существует. Если задан аргумент **-l , -list**, то **rc-service** выводит все доступные службы.

-e , -exists возвращает 0, если утилите удаётся найти запрошенную службу, иначе возвращает -1. **-r , -resolve** ведет себя аналогично, а также выводит полный путь службы в окне терминала.

СМ. ТАКЖЕ

[rc\(8\)](#), [stdout\(3\)](#)

АВТОРЫ

- Roy Marples roy@marples.name\

ПЕРЕВОД

- Елена Гаврилова e.vl.gavrilova@yandex.ru\

RC-STATUS

НАЗВАНИЕ

rc-status - отображение информации об уровнях запуска

СИНТАКСИС

rc-status [-aclsuC] [уровень запуска]

ОПИСАНИЕ

Утилита **rc-status** собирает и отображает информацию о состоянии служб на разных уровнях запуска. По умолчанию выводятся сведения о текущем уровне запуска и всех неназначенных службах, которые не остановлены, но при необходимости можно легко запросить и любой другой уровень.

Доступны следующие опции:

-a , -all

Отображать все уровни запуска и назначенные им службы.

-c , -crashed

Вывести полный перечень аварийно завершенных служб.

-l , -list

Вывести полный перечень определенных в системе уровней запуска.

-r , -runlevel

Сообщить название текущего уровня запуска.

-s , -servicelist

Вызвести список всех служб.

-u , -unused

Вызвести перечень служб, не назначенных ни одному уровню запуска.

-C , -nocolor

Отключить цветной вывод.

уровень запуска

Вызвать сведения только об указанном в аргументе *уровне запуска*.

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

rc-status возвращает 0, за исключением тех случаев, когда при проверке аварийно завершенных служб не найдено ни одной.

ПРИМЕЧАНИЯ О РЕАЛИЗАЦИИ

Если доступно дерево зависимостей, **rc-status** пытается вывести перечень служб на каждом уровне запуска в порядке текущих разрешенных зависимостей.

СМ. ТАКЖЕ

[rc\(8\)](#), [rc-update\(8\)](#)

АВТОРЫ

- Roy Marples [roy@marples.name](mailto:roy@marples.name)\

ПЕРЕВОД

- Елена Гаврилова [e.vl.gavrilova@yandex.ru](mailto:e.vl.gavrilova@yandex.ru)\

RC-UPDATE

НАЗВАНИЕ

rc-update - добавление/удаление служб на том или ином уровне запуска

СИНТАКСИС

rc-update [-s , -stack] add служба [уровень запуска ...] **rc-update [-s , -stack] delete** служба [уровень запуска ...] **rc-update [-u , -update] [-v , -verbose] show** [уровень запуска ...]

ОПИСАНИЕ

OpenRC использует именованные уровни запуска. **rc-update** позволяет быстро добавлять и удалять службы на различных уровнях запуска, не прибегая к редактированию конфигурационных файлов или работе с каталогом символических ссылок. Все службы должны располагаться в каталогах /etc/init.d или /usr/local/etc/init.d. Они должны также соответствовать стандарту сценариев запуска OpenRC.

add служба

Добавляет службу на указанный аргументом уровень запуска или, если команда дана без аргумента, на текущий уровень. Службы, добавляемые на уровень запуска boot, должны присутствовать в каталоге /etc/init.d.

delete служба

Удаляет службу с указанного аргументом уровня запуска или, если команда дана без аргумента, с текущего уровня.

show

Отображает все включенные службы и уровни запуска, которым они принадлежат. Если вы передаете этой команде аргумент, то в вывод будут включены только службы, которые запускаются на указанном уровне/-ях запуска.

-v , -verbose

Выводит все службы.

-u , -update

Принудительно обновляет кэш дерева зависимостей. Это может потребоваться при сбое часов (системные часы отстают от времени файла в каталоге /etc).

Если вам необходимо добавить или удалить службу на определенном уровне запуска, используйте опцию **-s , -stack**. Это делает возможным наследование уровней запуска.

СМ. ТАКЖЕ

[rc\(8\)](#), [rc-status\(8\)](#)

АВТОРЫ

- Roy Marples [roy@marples.name](mailto:roy@marples.name)\

- Сгенерировано kio_man, KDE, версия 4.7.2 (4.7.2)

ПЕРЕВОД

- Елена Гаврилова [\[e.vl.gavrilova@yandex.ru\]\(mailto:e.vl.gavrilova@yandex.ru\)\](mailto:e.vl.gavrilova@yandex.ru)

RUNSCRIPT

НАЗВАНИЕ

runscript - интерпретатор сценариев оболочки для служб

СИНТАКСИС

runscript [-D , -nodeps] [-d , -debug] [-s , -ifstarted] [-Z , -dry-run] [команда ...]

ОПИСАНИЕ

runscript фактически представляет собой интерпретатор shell-схемариев, предоставляющий простой интерфейс для зачастую сложных системных команд и демонов. Когда какая-либо служба вызывает команду, она сначала загружает множественный конфигурационный файл, затем управляющий им конфигурационный файл, далее /etc/rc.conf и, наконец, сам сценарий. Здесь **runscript** запускает заданную команду.

Команды определяются в сценарии как функции оболочки. Ниже приведен перечень функций, по умолчанию доступных для всех сценариев запуска:

describe

Описывает поведение службы и каждую команду, определяемую ей.

start

Прежде всего необходимо убедиться, что все службы, от которых зависит нормальная работа, запущены. Если какие-либо необходимые службы выдают ошибку при запуске, будет выполнен выход и выведено соответствующее сообщение; если же всё благополучно, вызывается функция запуска, если она существует.

stop

Прежде всего необходимо убедиться, что все службы, которые зависят от текущих операций, остановлены. Если какие-либо из этих служб выдают ошибку при остановке, будет выполнен выход и выведено соответствующее сообщение; если же всё благополучно, вызывается функция остановки, если она существует.

restart

Останавливает, а затем вновь запускает службу, вместе с зависимыми от нее.

status

Отображает текущее состояние службы. Будет возвращен соответствующий ему код, за исключением состояния "запущено" (started), при котором возвращается 0, для соответствия стандартному поведению команд.

zap

Устанавливает состояние службы "остановлено" (stopped) и удаляет все сохраненные данные о ней.

Следующие опции определяют параметры запуска службы:

-d , -debug

Включить xtrace в оболочке для отладки.

-D , -nodeps

Полностью игнорировать информацию о зависимостях, предоставляемую службой.

-s , -ifstarted

Выполнить команду, только если служба была запущена.

-q , -quiet

Отключить весь информационный вывод, генерируемый службой. Это не влияет на вывод любых других команд, не связанных с OpenRC.

-v , -verbose

Показывать дополнительный информационный вывод, генерируемый службой.

-Z , -dry-run

Показать, какие службы будут остановлены и/или запущены, но не останавливать / не запускать их.

Следующие переменные определяют сценарий службы:

_extra_commands_

Перечень дополнительных команд, определяемых службой, через пробел.

_extra_started_commands_

Перечень дополнительных команд, определяемых службой, через пробел. Работает только в том случае, если служба уже запущена.

_extra_stopped_commands_

Перечень дополнительных команд, определяемых службой, через пробел. Работает только в том случае, если служба уже остановлена.

description

Строка описания службы.

description\\$command_

Строка, описывающая дополнительную команду.

_start_stop_daemon_args_

Список аргументов, передаваемых start-stop-daemon'у при запуске службы.

command

Демон, обеспечивающий запуск и остановку посредством start-stop-daemon, если функция запуска-остановки не определена службой.

_command_args_

Список аргументов, передаваемых демону при запуске.

pidfile

Файл идентификатора процесса, используемый предыдущей командой.

name

Имя, используемое предыдущей командой.

retry

Количество попыток, которые будут предприняты при остановке службы. Это может быть как ожидание в секундах, так и многократные пары сигнал/ожидание (аналогично SIGTERM/5).

ЗАВИСИМОСТИ

Чтобы **runscript** запускался и останавливался в нужный момент по отношению к другим службам, вы должны определить **функцию зависимости**. Поскольку это функция, **depend** допускает самую тонкую настройку (см. пример ниже). Перечислим здесь функции, которые могут выступать в качестве **функции зависимости**. Вы просто передаете им имена служб, которые следует добавить к данному типу зависимости, а для удаления какой-либо службы сопровождаете ее префиксом !.

need

Служба не будет запускаться до запуска необходимых служб и не прекратит работу, пока службы, которым она необходима, не будут остановлены.

use

Служба попытается запустить другие используемые службы, добавленные на уровень запуска.

after

Служба будет запущена после данных служб и остановлена перед их остановкой.

before

Служба будет запущена перед запуском данных служб и остановлена после их остановки.

provide

Обеспечить данную виртуальную службу. Например, named обеспечивает dns.

config

Необходимо заново вычислить зависимости, поскольку файлы изменились.

keyword

Пометить службу ключевым словом. На данный момент поддерживаются следующие ключевые слова:

-shutdown

Не останавливать службу при выходе из системы. Как правило, оставшиеся демоны могут получать сигнал SIGTERM непосредственно перед конечным закрытием. Это ключевое слово обычно имеют службы, обеспечивающие работу в сети, такие как сценарии запуска network и dhcpcd.

-stop

Не останавливать службу при переходе на следующий уровень запуска, даже если его нет. Для выхода из системы.

-timeout

Другие службы должны ожидать вплоть до запуска службы. Используйте данное ключевое слово, если запуск службы может занять более минуты.

-jail

При виртуализации полностью исключать службу из зависимостей. Ее можно будет запустить, вызвав напрямую. Настройка через переменную **rc_sys** в файле /etc/rc.conf

-lxc

Аналогично -jail, но для виртуализации на уровне операционной системы (LXC, Linux Resource Containers).

-openvz

Аналогично -jail, но для систем OpenVZ.

-prefix

Аналогично -jail, но для систем Prefix.

-uml

Аналогично -jail, но для систем UML.

-vserver

Аналогично -jail, но для систем VServer.

-xen0

Аналогично -jail, но для систем Xen DOM0.

-xenu

Аналогично `-jail`, но для систем Xen DOMU.

О том, как переопределять зависимости с помощью конфигурационных файлов, см. раздел **ФАЙЛЫ**.

ВСТРОЕННЫЕ ФУНКЦИИ

`runscript` определяет некоторые встроенные функции, к которым вы можете обращаться из ваших служебных сценариев:

einfo [*строка*]

Выводит зеленый астериск перед строкой.

ewarn [*строка*]

Выводит желтый астериск перед строкой.

eerror [*строка*]

Выводит красный астериск перед строкой.

ebegin [*строка*]

Аналогично `einfo`, но с многоточием в конце.

eend *возвращаемая\величина* [*строка_*]

Если *возвращаемая\величина* не равна 0, выводит строку с `eerror` и `!!` в квадратных скобках в конце. Иначе выводить `ok` в квадратных скобках в конце строки. Возвращается значение *возвращаемой_величины_*.

ewend *возвращаемая\величина* [*строка_*]

Аналогично `eend`, но вместо `eerror` использовать `ewarn`.

Перечисленные команды могут иметь префикс `v`: в этом случае они будут действовать, только если переменная окружения `_EINFO_VERBOSE_` имеет значение `true`.

ewaitfile *время* *файл1* *файл2* ...

Ожидать время в секундах, пока не будут перебраны все файлы. Если все файлы существуют, возвращает 0, иначе значение, отличное от единицы. Если значение времени меньше 1, ожидание будет бесконечным.

is_newer_than *файл1* *файл2* ...

Если файл1 новее файла2, возвращает 0, иначе 1. Если файл2 - каталог, проверяет также его содержимое.

is_older_than *файл1* *файл2* ...

Если файл1 новее файла2, возвращает 0, иначе 1. Если файл2 - каталог, проверяет также его содержимое.

service_set_value *значение\имени_*

Сохраняет *значение\имени_* для дальнейшего восстановления. Когда служба прекратит свою работу, сохраненные значения будут утрачены.

service_get_value *имя*

Возвращает сохраненное значение под *именем*.

service_started [*служба*]

Если служба запущена, возвращает 0, иначе 1.

service_starting [*служба*]

Если служба запускается, возвращает 0, иначе 1.

service_inactive [*служба*]

Если служба неактивна, возвращает 0, иначе 1.

service_stopping [*служба*]

Если служба прекращает свою работу, возвращает 0, иначе 1.

service_stopped [*служба*]

Если служба остановлена, возвращает 0, иначе 1.

service_coldplugged [*служба*]

При холодном запуске службы возвращает 0, иначе 1.

service_wasinactive [*служба*]

Если служба была неактивна, возвращает 0, иначе 1.

service_started_daemon [*служба*] *демон* [*индекс*]

Если служба запустила демон через **start-stop-daemon**, возвращает 0, иначе 1. Если был указан индекс, это должен быть pid демона, запущенного службой.

mark_service_started [*служба*]

Пометить службу как запущенную.

mark_service_starting [*служба*]

Пометить службу как начинаяющую работу.

mark_service_inactive [*служба*]

Пометить службу как неактивную.

mark_service_stopping [*служба*]

Пометить службу как завершающую работу.

mark_service_stopped [*служба*]

Пометить службу как остановленную.

mark_service_coldplugged [*служба*]

Пометить службу как службу с холодной загрузкой.

mark_service_wasinactive [*служба*]

Пометить службу как неактивную.

checkpath

[-d , -directory] [-f , -file] [-m , -mode режим] [-o , owner владелец] путь ...

Проверяет существование *пути*, его типа, его *владельца*, *режимы доступа*. Если при проверке обнаруживается ошибка, путь будет исправлен.

yesno значение

Если значение соответствует YES, TRUE, ON или 1, возвращает 0 независимо от регистра, иначе возвращает 1.

ОКРУЖЕНИЕ

runscript присваивает значения следующим переменным окружения, которые можно использовать в сценариях служб:

_RC_SVCNAME_

Имя службы.

_RC_RUNLEVEL_

Текущий уровень запуска, на котором находится гс.

_RC_BOOTLEVEL_

Выбран загрузочный уровень запуска. По умолчанию - boot.

_RC_DEFAULTLEVEL_

Выбран основной уровень запуска. По умолчанию - default.

_RC_SYS_

Специальная переменная для дополнительного описания системы. Может принимать значения OPENVZ, XENU, XEN0, UML и VSERVER.

_RC_UNAME_

Результат выполнения команды `uname -s`.

ФАЙЛЫ

Файлы конфигурации, связанные с расположением службы. Если уже существует файл, оканчивающийся на .\${RC_RUNLEVEL}, будет использован он.

./conf.d/\${RC_SVCNAME}%.%.*}

Множественный конфигурационный файл. Например: если \${RC_SVCNAME} - net.eth1, обращаться к ./conf.d/net.

./conf.d/\${RC_SVCNAME}

Конфигурационный файл службы.

/etc/rc.conf

Конфигурационный файл хоста.

За исключением /etc/rc.conf, конфигурационные файлы могут также переопределять зависимости служб с помощью переменных. Просто добавьте к зависимости префикс rc_. Примеры:

```
# В то время как большинство служб не требуют определенного интерфейса,
# наша
# конфигурация openvpn в таком нуждается, а именно в bge0.
rc_need="net.bge0"
# В файле /etc/rc.conf пропишем
rc_openvpn_need="net.bge0"

# Службы не должны зависеть от интерфейса tap1 для сетевой работы,
# но нам необходимо добавить net.tap1 к уровню запуска default.
rc_provide="!net"
# В файле /etc/conf.d/net пропишем
rc_provide_tap1="!net"
# В файле /etc/rc.conf пропишем
rc_net_tap1_provide="!net"

# Ключевые слова можно использовать с отрицанием. Это особенно удобно для
# пользователей prefix,
# тестирующих OpenRC.
rc_keyword="!noprefix"
```

ПРИМЕРЫ

Ниже приводится пример сценария службы для foo.

```
#!/sbin/runscript
command=/usr/bin/foo
command_args="${foo_args} --bar"
```

```

pidfile=/var/run/foo.pid
name="FooBar Daemon"

description="FooBar is a daemon that eats and drinks"
extra_commands="show"
extra_started_commands="drink eat"
description_drink="Opens mouth and reflexively swallows"
description_eat="Chews food in mouth"
description_show="Shows what's in the tummy"

_need_dbus()
{
    grep -q dbus /etc/foo/plugins
}

depend()
{
    # Мы сохраняем файл идентификатора процесса и пишем в каталог
    /var/cache, поэтому нам потребуется localmount:
    need localmount
    # Мы можем использовать сеть, но это не обязательно:
    use net
    # Служба должна следовать за bootmisc, чтобы каталог /var/run
    инициализировался до того,
    # как мы поместим туда файл идентификатора процесса:
    after bootmisc

    # Foo может использовать демон dbus.
    # Но если бы добавим dbus, пока foo работает,
    # и потом остановим dbus, нам не нужно останавливать foo, поскольку
foo не использовала dbus:
    config /etc/foo/plugins
    local _need=
    if service_started; then
        _need=`service_get_value need`
    else
        if _need_dbus; then
            _need="${_need} dbus"
        fi
    fi
    need ${_need}
}

start_pre()
{
    # Убедимся, что наши каталоги не содержат ошибок:
    checkpath --dir --owner foo:foo --mode 0664 \
        /var/run/foo /var/cache/foo
}

start_post()
{
    # Сохраним необходимую службу:
    if _need_dbus; then
        service_set_value need dbus
    fi
}

```

```

}

stop_post() {
    # Удалим мусор:
    rm -rf /var/cache/foo/*
}

drink()
{
    ebegin "Starting to drink"
    ${command} --drink beer
    eend $? "Failed to drink any beer :("
}

eat()
{
    local result=0 retval= ate= food=
    ebegin "Starting to eat"

    if yesno "${foo_diet}"; then
        eend 1 "We are on a diet!"
        return 1
    fi

    for food in /usr/share/food/*; do
        veinfo "Eating `basename ${food}`"
        ${command} --eat ${food}
        retval=$?
        : $(( ${result} += ${retval} ))
        [ ${retval} = 0 ] && ate="${ate} `basename ${food}`"
    done

    if eend ${result} "Failed to eat all the food"; then
        service_set_value ate "${ate}"
    fi
}

show()
{
    einfo "Foo has eaten: `service_get_value ate`"
}

```

ОШИБКИ

По причинам, связанным со способом загрузки конфигурационных файлов и необходимостью обрабатывать несколько служебных каталогов, вы можете использовать в каталогах служб только символические ссылки на другие службы из того же каталога. Вы не можете создавать символическую ссылку, указывающую на службу из другого каталога, даже если другой служебный каталог.

`is_older_than` должна бы возвращать 0 при успешной отработке, однако в этом случае она возвращает 1, для совместимости с базовым набором (baselayout) Gentoo. Мы рекомендуем пользователям использовать функцию `is_newer_`, которая работает привычным образом.

СМ. ТАКЖЕ

[einfo\(3\)](#), [rc\(8\)](#), [rc-status\(8\)](#), [rc-update\(8\)](#), [rc_plugin_hook\(3\)](#), [sh\(1p\)](#), [start-stop-daemon\(8\)](#), [uname\(1\)](#)

АВТОРЫ

- Roy Marples roy@marples.name\

ПЕРЕВОД

- Елена Гаврилова e.vl.gavrilova@yandex.ru\

START-STOP-DAEMON

НАЗВАНИЕ

start-stop-daemon - обеспечение запуска и остановки демонов

СИНТАКСИС

start-stop-daemon **-S** , **-start** демон [--] [аргументы] **start-stop-daemon** **-K** , **-stop** демон **start-stop-daemon** **-s** , **-signal** *signal* демон

ОПИСАНИЕ

Утилита **start-stop-daemon** обеспечивает надежный метод запуска и остановки демонов, а также передачи им сигнала. Если не используется ни опция **-K** , **-stop**, ни **-s** , **-signal**, подразумевается, что демон необходимо запустить. Если демон не запускается самостоятельно в фоновом режиме и не создает файл идентификатора процесса, это может безопасно выполнить **start-stop-daemon**.

Если **start-stop-daemon** используется для той или иной службы OpenRC, OpenRC может проверить, работает ли демон. Если нет, то служба помечается как аварийно остановленная.

Ниже приводятся опции, служащие для указания демона и способа его запуска или остановки:

-x , **-exec** демон

Запускаемый или останавливаемый демон. Если эта опция не указана, то будет использован первый аргумент вне опций.

-p , **-pidfile** файл идентификатора процесса

При запуске демона он за разумное время должен создать рабочий файл идентификатора процесса. При остановке будут остановлены только процессы, перечисленные в файле идентификатора.

-n , **-name** имя

Определяет демон по имени процесса, а не по файлу идентификатора или по имени исполняемого файла.

-i , **-interpreted**

При выборе процесса по имени необходимо убедиться, что ему соответствует подходящий интерпретатор. Как только, например, запускается демон foo, **start-stop-daemon** ищет процесс. Если интерпретируемый демон изменяет свое имя процесса, это не будет работать.

-u , **-user** пользователь [:группа]

Запускает демон от пользователя и соответственно обновляет значение \$HOME или останавливает демоны, принадлежащие пользователю. При желании вы можете добавить и группу.

-t , **-test**

Только отображает сведения о действии/-ях, которые должны быть выполнены, ничего не делая. Возвращается то же значение, что и в случае действительно запуска и отработки команды.

-v , **-verbose**

Отображает сведения о действии/-ях непосредственно перед его/их выполнением.

-P , **-progress**

Отображает ход выполнения в консоли: одна точка обозначает секунду ожидания.

Следующие опции используются только при запуске демонов:

-a , -startas имя

Заменяет имя процесса демона на указанное *имя*. Это заменит лишь первый аргумент, передаваемый демону.

-b , -background

Принудительно запускает демон в фоновом режиме. Некоторые демоны не создают идентификационных файлов - их удобно запускать в фоне и использовать эту опцию в связке с **-m , -make-pidfile** для создания рабочего файла идентификатора.

-d , -chdir путь

Перед запуском демона изменяет рабочий каталог на указанный.

-r , -chroot путь

Перед запуском демона выполняет chroot в каталоге по указанному *пути*. Другие пути, например, путь доступа к демону, адрес нового корневого каталога root и файла идентификатора процесса, должны быть относительными к chroot.

-c , -chuid пользователь

Аналогично опции **-u , -user**.

-e , -env VAR=VALUE

Присваивает переменной окружения VAR значение VALUE.

-g , -group группа

Запускает демон в группе.

-k , -umask режим

Задает маску файла демона.

-m , -make-pidfile

Сохраняет идентификатор процесса демона в файле, указанном в опции **-p , -pidfile**. Имеет смысл только применительно к демонам, запущенным в основном режиме, которые принудительно переводятся в фоновый режим опцией **--b , -background**.

-I , -ionice класс [:данные]

Изменяет приоритет ввода-вывода для демона. Аргумент *класс* может иметь значение 0 - нет, 1 - real time (приоритетный доступ), 2 - best effort (приоритет определяется планировщиком) и 3 - idle (доступ только тогда, когда другие процессы не требуют ввода/вывода). Аргумент *данные* может принимать значения от 0 до 7 включительно.

-N , -nice уровень

Изменяет приоритет запуска демона.

-1 , -stdout журнал

При запуске с опцией **-background** перенаправляет стандартный вывод процесса в журнал. В качестве аргумента необходимо указывать абсолютный путь доступа к файлу, но относительный к пути, опционально заданному опцией **-r , -chroot**. Журнал может быть и именованным каналом.

-w , -wait время

После запуска ожидать указанное *время* (в миллисекундах), затем проверить, продолжает ли демон работу. Эта опция удобна для демонов, которые проверяют конфигурацию после ветвления или устранения "гонок", когда файл идентификатора записывается после ветвления процесса.

-2 , -stderr журнал

Эта опция аналогична **-1**, **-stdout**, но со стандартным выводом ошибок.

Следующие опции используются только для остановки демонов:

-R , **-retry** пауза | сигнал / пауза

Вы можете указать либо продолжительность паузы в секундах, либо несколько пар сигнал/пауза для расписания остановки демонов. Если опция не задана, будет использовано значение по умолчанию SIGTERM/5.

ОКРУЖЕНИЕ

С помощью переменной `_SSD_NICELEVEL_` также можно устанавливать очередь запуска демонов, но приоритет будет иметь опция в командной строке.

Переменная `_SSD_STARTWAIT_` аналогична опции **-w**, **-wait**, описанной выше. `/etc/rc.conf`, **start-stop-daemon** ожидает проверки, продолжает ли демон работу.

ПРИМЕЧАНИЕ

Для синтаксического анализа опций **start-stop-daemon** использует `getopt(3)`: этот инструмент позволяет принимать опцию с префиксом `--`, останавливая обработку текущих опций на данном этапе. Все последующие аргументы передаются демону, который запускает соответствующие службы, и используются при обнаружении демона, который необходимо остановить или которому необходимо передать сигнал.

СМ. ТАКЖЕ

`chdir(2)`, `chroot(2)`, `getopt(3)`, `nice(2)`, `rc_find_pids(3)`

ОШИБКИ

Не обращаясь к файлу идентификатора процесса, **start-stop-daemon** не может остановить интерпретируемый демон, который уже не существует.

ИСТОРИЯ

Впервые **start-stop-daemon** был использован в Debian.

Настоящая реализация стала результатом полной переработки изначальной версии: процесс ищет код в библиотеке OpenRC (`librc`, `-lrc`), чтобы к нему могли обращаться другие программы.

АВТОРЫ

- Roy Marples [\[roy@marples.name\]\(mailto:roy@marples.name\)\](mailto:roy@marples.name)

ПЕРЕВОД

- Елена Гаврилова [\[e.vl.gavrilova@yandex.ru\]\(mailto:e.vl.gavrilova@yandex.ru\)\](mailto:e.vl.gavrilova@yandex.ru)

Справка по portage-utils

Portage-utils ([app-portage/portage-utils](#)) - набор легких и быстрых утилит для извлечения информации о пакетах в Portage; написаны на C.

- [qatom](#) - сравнение строк атомов
- [qcachе](#) - поиск в кэше метаданных
- [qcheck](#) - проверка целостности установленного пакета
- [qdepends](#) - вывод информации о зависимостях
- [qfile](#) - вывод всех пакетов, которым принадлежит файл
- [qgrep](#) - поиск по шаблону в ебилдах
- [qlist](#) - вывод файлов, принадлежащих данному пакету
- [qlop](#) - анализатор журналов установки

- [qmerge](#) - скачивание и установка бинарного пакета
- [qpkg](#) - информация о бинарном пакете Gentoo
- [qsearch](#) - поиск по шаблону пакетов с описанием
- [qsize](#) - данные по размеру пакетов
- [qtbz2](#) - работа с tbz2-пакетами
- [quse](#) - узнать, какие пакеты используют USE-флаг
- [qxpak](#) - работа с архивами xpak

qatom

НАЗВАНИЕ

qatom - сравнение строк атомов

СИНТАКСИС

qatom <опции> <пакет>

ОПИСАНИЕ

Доступные опции: **-[cvqChV]**

-c, --compare

* Сравнить два атома

-v, --verbose

* Предлагать подробный вывод

-q, --quiet

* Предлагать компактный вывод - в частности, не выводить предупреждения

-C, --nocolor

* Не использовать цветной вывод

-h, --help

* Вывести эту справку и выйти

-V, --version

* Вывести версию программы и выйти

\$Id: qatom.1,v 1.15 2007/05/11 08:25:06 solar Exp \$

ПЕРЕВОД

- Елена Гаврилова [\(mailto:e.vl.gavrilova@yandex.ru\)](mailto:e.vl.gavrilova@yandex.ru)

Май 2007

qcache

НАЗВАНИЕ

qcache - поиск в кэше метаданных

СИНТАКСИС

qcache <опции> <действие> <аргументы>

ОПИСАНИЕ

Доступные версии: -[p:c:idtansvqChV]

-p, --matchpkg <аргумент>

* Искать соответствие по названию пакета

-c, --matchcat <аргумент>

* Искать соответствие по имени категории

-i, --imlate

* Выводить пакеты, которые могут маркироваться как стабильные для данной архитектуры

-d, --dropped

* Выводить пакеты, которые после обновления версии не имеют ключевых слов для данной архитектуры

-t, --testing

* Выводить пакеты, имеющие версии, маркированные как ~arch, и не имеющие стабильных версий для данной архитектуры

-s, --stats

* Выводить статистику дерева портежей

-a, --all

* Выводить пакеты, у которых хотя бы одна версия имеет ключевое слово для данной архитектуры

-n, --not

* Выводить пакеты, которые не имеют ключевых слов для данной архитектуры

-v, --verbose

* Предлагать подробный вывод

-q, --quiet

* Предлагать компактный вывод - в частности, не выводить предупреждения

-C, --nocolor

* Не использовать цветной вывод

-h, --help

* Вывести эту справку и выйти

-V, --version

* Вывести версию программы и выйти

\$Id: qcache.1,v 1.6 2007/05/11 08:25:06 solar Exp \$

ПЕРЕВОД

- Елена Гаврилова [\(mailto:e.vl.gavrilova@yandex.ru\)\](mailto:e.vl.gavrilova@yandex.ru)

Май 2007

qcheck

НАЗВАНИЕ

qcheck - проверка целостности установленного пакета

СИНТАКСИС

qcheck <опции> <пакет>

ОПИСАНИЕ

Допустимые опции: -[еauАНТВvqChV]

-e, --exact

* Точное совпадение (только CAT/PN или PN без PV)

-a, --all

* Вывести все пакеты

-u, --update

* Обновить данные об отсутствующих файлах, контрольных суммах и времени последнего изменения пакетов

-A, --noafk

*忽略する既に存在しないファイル

-H, --nohash

*忽略する差異/存在しない контрольной суммы

-T, --nomtime

*忽略する差異 во времени последнего изменения файлов

-B, --badonly

* Выводить только пакеты, содержащие битые файлы, исключая /etc.

-v, --verbose

* Предлагать подробный вывод

-q, --quiet

* Предлагать компактный вывод - в частности, не выводить предупреждения

-C, --nocolor

* Не использовать цветной вывод

-h, --help

* Вывести эту справку и выйти

-V, --version

* Вывести версию программы и выйти

\$Id: qcheck.1,v 1.24 2008/04/12 17:26:00 solar Exp \$

ПЕРЕВОД

- Елена Гаврилова [\(mailto:e.vl.gavrilova@yandex.ru\)\](mailto:e.vl.gavrilova@yandex.ru)

Апрель 2008

qdepends

НАЗВАНИЕ

qdepends - вывод информации о зависимостях

СИНТАКСИС

qdepends <опции> <пакет>

ОПИСАНИЕ

Допустимые опции: -[drpaNk:Q:vqChV]

-d, --depend

* Показывать информацию по DEPEND (по умолчанию)

-r, --rdepend

* Показывать информацию по RDEPEND

-p, --pdepend

* Показывать информацию по PDEPEND

-k, --key <аргумент>

* Пользовательский ключ vdb

-Q, --query <аргумент>

* Запрашивать обратные зависимости

-N, --name-only

* Отображать только название пакета

-a, --all

* Показывать всю информацию о DEPEND

-v, --verbose

* Предлагать подробный вывод

-q, --quiet

* Предлагать компактный вывод - в частности, не выводить предупреждения

-C, --nocolor

* Не использовать цветной вывод

-h, --help

* Вывести эту справку и выйти

-V, --version

* Вывести версию программы и выйти

\$Id: qdepends.1,v 1.22 2007/05/11 08:25:06 solar Exp \$

ПЕРЕВОД

- Елена Гаврилова [\(mailto:e.vl.gavrilova@yandex.ru\)\](mailto:e.vl.gavrilova@yandex.ru)

Май 2007

qfile

НАЗВАНИЕ

qfile - вывод всех пакетов, которым принадлежит файл

СИНТАКСИС

qfile <файл>

ОПИСАНИЕ

Доступные опции: **-[ef:m:oRx:vqChV]**

-e, --exact

* Искать точное соответствие

-f, --from <аргумент>

* Брать аргументы из файла в <аргументе> ("-" для стандартного ввода)

-m, --max-args <аргумент>

* Обрабатывать файлы группами в количестве, заданном <аргументом> (значение по умолчанию - 5000)

-o, --orphans

* Вывести список файлов

-R, --root-prefix

* Считать, что префикс \$ROOT уже добавлен к аргументам

-x, --exclude <аргумент>

* Не рассматривать пакет в <аргументе>

-v, --verbose

* Предлагать подробный вывод

-q, --quiet

* Предлагать компактный вывод - в частности, не выводить предупреждения

-C, --nocolor

* Не использовать цветной вывод

-h, --help

* Вывести эту справку и выйти

-V, --version

* Вывести версию программы и выйти

\$Id: qfile.1,v 1.24 2007/01/13 19:17:40 Hanисано для Gentoo solar u vapier, gentoo.org

НАХОЖДЕНИЕ ПАКЕТОВ - ВЛАДЕЛЬЦЕВ ФАЙЛОВ

Это поведение утилиты **qfile** по умолчанию. Будет выведен перечень пакетов, которым принадлежат запрошенные файлы (вместо файлов могут быть каталоги, символические ссылки и любые другие элементы, устанавливаемые Portage). Для запроса можно использовать как путь, так и просто имя файла. Стандартный вывод утилиты включает названия пакетов и полные пути доступа к файлам, удовлетворяющим запросу. Если вы используете опцию **--exact**, будут отображаться также версии пакетов. Если же выставить опцию **--quiet**, будут выведены только названия пакетов, без путей доступа к файлам. Аналогично опции **--exact** работает **--verbose**, но выводит еще больше информации. Когда все пакеты, которым принадлежат файлы, указанные в аргументе, найдены, **qfile** возвращает 0.

Найти пакет(ы), которому/-ым принадлежит "/bin/bash":

```
$ qfile -q /bin/bash
app-shells/bash
```

Найти пакет(ы), которому/-ым принадлежит любой файл с именем "bash", и показать пути доступа к нему:

```
$ qfile bash
app-shells/bash (/bin/bash)
```

```
app-shells/bash (/etc/bash)
```

Найти пакет(ы), которому/-ым принадлежит любой файл с именем "bash", в текущем каталоге. Кроме того, будет выведен точный номер версии:

```
$ cd /bin  
$ qfile -e ./bash  
app-shells/bash-3.1_p17 (/bin/bash)
```

Найти пакет(ы), которому/-ым принадлежат библиотеки, требуемые бинарным Bash:

```
$ qfile $(scanelf -nq -F%n#F /bin/bash | tr , '\n')  
sys-libs/nurses (/lib/libncurses.so.5)  
sys-libs/glibc (/lib/libdl.so.2)  
sys-libs/glibc (/lib/libc.so.6)
```

ОБНАРУЖЕНИЕ НЕИСПОЛЬЗУЕМЫХ ФАЙЛОВ

qfile умеет также находить файлы, не принадлежащие ни одному пакету; для этого необходимо использовать опцию **--orphans**. Фактически при этом утилита осуществляет операцию, обратную своему обычному поведению: выводит список элементов, для которых не найдено соответствий в базе данных установленных пакетов. Опция **--exact** в данном режиме не работает, но вы можете использовать **--verbose**, если вы хотите видеть дополнительную информацию. Если же вы выставите опцию **--quiet**, вывод утилиты будет полностью отключен: команда отработает молча и вернет 0 только в том случае, если неиспользуемых файлов среди запрошенных вами нет.

Найти неиспользуемые файлы библиотек:

```
$ qfile -o $(find /lib /usr/lib -name "*.la")  
/usr/lib/libGL.la
```

Найти библиотеки, которые нужны для работы бинарному пакету "foo", но при этом не были установлены ни одним пакетом:

```
$ qfile -o $(scanelf -nq -F%n#F /путь/к/foo | tr , '\n')  
libinstalledmanually.so.2
```

ОБРАБОТКА ПЕРЕМЕННОЙ ROOT

Установив переменную окружения *ROOT*, вы можете указать **qfile**, в какой системе работать. В приведенном примере осуществляется поиск пакета-владельца файла "/bin/sh" - сначала в вашей основной системе, а затем в системе, смонтированной в каталог "/mnt":

```
$ qfile -q /bin/sh  
app-shells/bash  
$ ROOT=/mnt qfile -q /bin/sh  
sys-apps/busybox
```

Обратите внимание, что в обеих командах аргументов запроса является "/bin/sh": по умолчанию **qfile** ищет по путям доступа к файлам, прописанным в базе данных пакетов адресуемой системы, и эти пути не включают \$ROOT. Если же вы хотите искать по действительным путям доступа к файлам (включая точку монтирования), используйте опцию **--root-prefix (-R)**:

```
$ ROOT=/mnt qfile -Rq /mnt/bin/sh  
sys-apps/busybox
```

Кроме того, опция **-R** отличается от настроек по умолчанию выводом путей доступа. В предыдущем примере префикс **\$ROOT** не используется, а в примере ниже - используется:

```
$ ROOT=/mnt qfile sh  
sys-apps/busybox (/bin/sh)  
$ ROOT=/mnt qfile -R sh  
sys-apps/busybox (/mnt/bin/sh)
```

Разумеется, сказанное справедливо и для поиска неиспользуемых файлов:

```
$ ROOT=/mnt qfile -o $(ls /mnt/bin/ | sed 's:^/mnt::')  
/bin/dostuff.sh  
$ ROOT=/mnt qfile -Ro /mnt/bin/*  
/mnt/bin/dostuff.sh
```

СЧИТЫВАНИЕ АРГУМЕНТОВ ИЗ ФАЙЛА (СТАНДАРТНЫЙ ВВОД)

Пытаясь запустить **qfile** с большим количеством аргументов, вы можете столкнуться со следующей ошибкой интерпретатора:

```
$ qfile -o $(find /usr/lib)  
bash: ./qfile: Argument list too long
```

Во избежание этого следует использовать опцию **--from (-f)**, которая позволяет считывать список аргументов из файла:

```
$ find /usr/lib > ~/usr-lib.list  
$ qfile -o -f ~/usr-lib.list  
/usr/lib/libMagick-5.5.7-Q16.so.0.0.0  
/usr/lib/libGL.so  
...
```

В каждой строке списка аргументов должен располагаться только один файл, без какого либо мусора (без пробелов в начале или в конце строки, без пропусков строки и т.п.). Стандартный формат вывода **find**, как правило, вполне удобен.

Но аргументы могут быть перенаправлены и со стандартного ввода, с помощью псевдо-имени файла **"-"**; это удобно при использовании конвейера:

```
$ find /usr/lib | qfile -o -f -  
/usr/lib/libMagick-5.5.7-Q16.so.0.0.0  
/usr/lib/libGL.so  
...
```

Ниже приведен пример сценария, которые ищет некоторые файлы, которые могут оказаться неиспользуемыми конфигурационными файлами, не удаленными Portage при удалении или обновлении пакетов:

```
#!/bin/bash  
SEARCH_PATHS=$(portageq envvar CONFIG_PROTECT)  
SEARCH_MASK=$(portageq envvar CONFIG_PROTECT_MASK)  
/etc/runlevels /etc/portage  
/etc/ssl/certs /etc/ssh  
/etc/bash_completion.d /etc/cups"  
for path in ${SEARCH_MASK} ; do  
    EXCLUDE="$EXCLUDE" -not -path ${path}/*"
```

```

done
set -f
find ${SEARCH_PATHS} ${EXCLUDE} | qfile -o -f -

```

ВНИМАНИЕ: приведенный сценарий не следует рассматривать как рабочий инструмент - это не более чем набросок. Не удаляйте вслепую файлы, которые будут выведены при запуске этого сценария!

Если аргументычитываются из файла или перенаправляются со стандартного ввода, для улучшения производительности **qfile** обрабатывает данные блоками по 5000 элементов (будет осуществлен поиск пакетов-владельцев первых 5000 файлов, затем 5000 следующих и т.д.). В большинстве случаев это условное значение вполне удобно, но при желании вы можете изменить его с помощью опции **--max-args (-m)**. Увеличение значения по умолчанию увеличит объем используемой оперативной памяти, но может ускорить обработку запросов с очень большим количеством аргументов; тем не менее всякое необдуманное изменение может самым негативным образом оказаться на производительности. Не вдаваясь в детали, можно сказать, что вам, скорее всего, не потребуется изменять стандартное значение.

ПОИСК КОНФЛИКТУЮЩИХ ФАЙЛОВ

Последней опцией утилиты **qfile** является **--exclude (-x)**. Она позволяет пропустить при поиске пакетов-владельцев файлов один определенный пакет. Эта опция принимает один аргумент: это может быть название пакета (например, **bash** или **app-shells/bash**), название пакета с указанием версии (например, **bash-3.2_p9-r1** или **app-shells/bash-3.2_p9-r1**), название пакета с указанием слота (**bash:0** или **app-shells/bash:0**). Опция полезна для выявления конфликтов между пакетами (в частности, путем сопоставления содержимого одного пакета с другими).

Так, приведенный ниже сценарий предназначен для поиска конфликтов между любыми установленными пакетами. Имейте в виду, что это займет некоторое время:

```

#!/bin/bash
cd $(portageq vdb_path)
for pkg in *-*/*-* ; do
    [[ -f ${pkg}/CONTENTS ]] || continue
    collisions=$(sed -n
        '/^obj\|^sym/s:^... \([^\ ]+\).*:!\1:p'
        ${pkg}/CONTENTS
        | qfile -e -x ${pkg} -f -)
    [[ -n ${collisions} ]]
    && echo ">>> ${pkg}:"
    && echo "${collisions}"
done

```

А этот сценарий может быть использован для проверки, не имеет ли бинарный пакет (.tbz2-архив) конфликтов с каким-либо из установленных пакетов, за исключением того, который он может заменить (с тем же именем и слотом), если таковой имеется:

```

#!/bin/bash
pkgver=$(basename "${1}")
pkgver=${pkgver%.*.tbz2}
pn=$(qatom ${pkgver} | cut -d\ -f2)
tmpdir=$(mktemp -t -d) || exit 1
tarbz2=${tmpdir}/${pkgver}.tar.bz2
xpak=${tmpdir}/${pkgver}.xpak
qtbz2 -s "${1}" "${tarbz2}" "${xpak}"
categ=$(qxpak -0 -x "${xpak}" CATEGORY)
slot=$(qxpak -0 -x "${xpak}" SLOT)
tar tjf "${tarbz2}"
| sed -e 's:^\./::' -e '\:$:d'
| qfile -e -f - -x ${categ}/${pn}:${slot}

```

```
rm -rf "${tmpdir}"
```

РАЗРАБОТАНО ПРИ УЧАСТИИ

- TGL degrenier@easyconnect.fr

ПЕРЕВОД

- Елена Гаврилова [\(mailto:e.vl.gavrilova@yandex.ru\)](mailto:e.vl.gavrilova@yandex.ru)

Январь 2007

qgrep

НАЗВАНИЕ

qgrep - поиск по шаблону в ебилдах

СИНТАКСИС

qgrep <различные аргументы>

ОПИСАНИЕ

Доступные опции: **-[iHNclLexJEsS:B:A:vqChV]**

-I, --invert-match

* Выбрать все строки, кроме совпадающих с шаблоном

-i, --ignore-case

*忽略する

-H, --with-filename

* Выводить имя файла при каждом совпадении

-N, --with-name

* Выводить имя пакета или екласса при каждом совпадении

-c, --count

* Выводить только количество строк на ФАЙЛ, совпадающих с шаблоном

-l, --list

* Выводить только имена ФАЙЛОВ, совпадающие с шаблоном

-L, --invert-list

* Выводить только имена ФАЙЛОВ, не совпадающие с шаблоном

-e, --regexp

* Использовать как ШАБЛОН регулярное выражение

-x, --extended

* Использовать как ШАБЛОН расширенное регулярное выражение

-J, --installed

* Искать не в дереве, а в установленных ебилдах

-E, --eclass

* Искать не в ебилдах, а в еклассах

-s, --skip-comments

* Пропускать строки комментариев
-S, --skip <аргумент>
* Пропускать строки, совпадающие с шаблоном в <аргументе>
-B, --before <аргумент>
* Выводить строки, содержащие <аргумент> в начале
-A, --after <аргумент>
* Выводить строки, содержащие <аргумент> в конце
-v, --verbose
* Предлагать подробный вывод
-q, --quiet
* Предлагать компактный вывод - в частности, не выводить предупреждения
-C, --nocolor
* Не использовать цветной вывод
-h, --help
* Вывести эту справку и выйти
-V, --version
* Вывести версию программы и выйти
\$Id: qgrep.1,v 1.17 2007/04/05 18:25:27 solar Exp \$

ПЕРЕВОД

- Елена Гаврилова [\(mailto:e.vl.gavrilova@yandex.ru\)\](mailto:e.vl.gavrilova@yandex.ru)

Апрель 2007

qlist

НАЗВАНИЕ

qlist - вывести файлы, принадлежащие данному пакету

СИНТАКСИС

qlist <опции> <пакет>

ОПИСАНИЕ

Доступные опции: **-[ISULDeadosvqChV]**

-I, --installed

* Показывать только установленные пакеты

-S, --slots

* Отображать установленные пакеты со слотами

-L, --separator

* Использовать : в качестве разделителя слота

-U, --umap

* Отображать установленные пакеты с используемыми флагами

-D, --dups

* Показывать только пакеты с копиями

-e, --exact

* Точное соответствие (только CAT/PN или PN без PV)

-a, --all

* Показывать все установленные пакеты

-d, --dir

* Показывать только каталоги

-o, --obj

* Показывать только объекты

-s, --sym

* Показывать только символические ссылки

-v, --verbose

* Предлагать подробный вывод

-q, --quiet

* Предлагать компактный вывод - в частности, не выводить предупреждения

-C, --nocolor

* Не использовать цветной вывод

-h, --help

* Вывести эту справку и выйти

-V, --version

* Вывести версию программы и выйти

\$Id: qlist.1,v 1.24 2007/09/08 06:31:48 solar Exp \$

ПЕРЕВОД

- Елена Гаврилова [\[e.vl.gavrilova@yandex.ru\]\(mailto:e.vl.gavrilova@yandex.ru\)\](mailto:e.vl.gavrilova@yandex.ru)

Сентябрь 2007

qlop

НАЗВАНИЕ

qlop - анализатор журналов установки

СИНТАКСИС

qlop <опции> <пакет>

ОПИСАНИЕ

Доступные опции: **-[gtHluscf:vqChV]**

-g, --gauge

* Определить, сколько раз устанавливался пакет

-t, --time

* Рассчитать среднее время установки пакета

-H, --human

* Выводить секунды в удобном для восприятия виде (требует опции **-t**)

-l, --list

* Показывать историю установок

-u, --unlist

* Показывать историю удалений

-s, --sync

* Показывать историю синхронизаций

-c, --current

* Показывать, какие пакеты сейчас устанавливаются

-f, --logfile <аргумент>

* Обращаться к журналу emerge вместо /var/log/emerge.log

-v, --verbose

* Предлагать подробный вывод

-q, --quiet

* Предлагать компактный вывод - в частности, не выводить предупреждения

-C, --nocolor

* Не использовать цветной вывод

-h, --help

* Вывести эту справку и выйти

-V, --version

* Вывести версию программы и выйти

\$Id: qlop.1,v 1.20 2007/05/11 08:25:06 solar Exp \$

ПЕРЕВОД

- Елена Гаврилова [\[e.vl.gavrilova@yandex.ru\]\(mailto:e.vl.gavrilova@yandex.ru\)](mailto:e.vl.gavrilova@yandex.ru)

Май 2007

qmerge

НАЗВАНИЕ

qmerge - скачивание и установка бинарного пакета

СИНТАКСИС

qmerge <опции> <пакеты>

ОПИСАНИЕ

Доступные опции: **-[fFsKUpuyO5vqChV]**

-f, --fetch

* Скачать пакет и последние метаданные пакетов

-F, --force

* Скачать пакет (минуя метаданные)

-s, --search

* Искать среди доступных пакетов
-K, --install
* Установить пакет
-U, --unmerge
* Удалить пакет
-p, --pretend
* Симулировать действие
-u, --update
* Только обновить
-y, --yes
* Не спрашивать перед перезаписью
-O, --nodeps
* Не устанавливать зависимости
-5, --nomd5
* Не проверять MD5-хеш файлов
-v, --verbose
* Предлагать подробный вывод
-q, --quiet
* Предлагать компактный вывод - в частности, не выводить предупреждения
-C, --nocolor
* Не использовать цветной вывод
-h, --help
* Вывести эту справку и выйти
-V, --version
* Вывести версию программы и выйти

\$Id: qmerge.1,v 1.14 2007/05/11 08:25:06 solar Exp \$

ПЕРЕВОД

- Елена Гаврилова [\(mailto:e.vl.gavrilova@yandex.ru\)](mailto:e.vl.gavrilova@yandex.ru)

Май 2007

qpkg

НАЗВАНИЕ

qpkg - информация о бинарном пакете Gentoo

СИНТАКСИС

qpkg <опции> <разные аргументы>

ОПИСАНИЕ

Доступные опции: **-[cpP:vqChV]**

-c, --clean

- * Очистить каталог пакета от неиспользуемых бинарных файлов
- p, --pretend**
- * Симулировать действие
- P, --pkgdir <аргумент>**
- * Альтернативный каталог пакетов
- v, --verbose**
- * Предлагать подробный вывод
- q, --quiet**
- * Предлагать компактный вывод - в частности, не выводить предупреждения
- C, --nocolor**
- * Не использовать цветной вывод
- h, --help**
- * Вывести эту справку и выйти
- V, --version**
- * Вывести версию программы и выйти

\$Id: qpkg.1,v 1.19 2007/06/01 00:10:07 solar Exp \$

ПЕРЕВОД

- Елена Гаврилова [\(mailto:e.vl.gavrilova@yandex.ru\)](mailto:e.vl.gavrilova@yandex.ru)

Май 2007

qsearch

НАЗВАНИЕ

qsearch - поиск по пакету/описанию пакета

СИНТАКСИС

qsearch <опция> <шаблон>

ОПИСАНИЕ

Доступные опции: **-[acsSNHvqChV]**

-a, --all

* Список описаний каждого пакета в кэше

-c, --cache

* Использовать кэш portage

-s, --search

* Поиск по шаблону среди названий пакетов

-S, --desc <аргумент>

* Поиск по шаблону среди описаний пакетов

-N, --name-only

* Показывать только название пакета.

-H, --homepage

* Показывать информацию о домашней странице программы

-v, --verbose

* Предлагать подробный вывод

-q, --quiet

* Предлагать компактный вывод - в частности, не выводить предупреждения

-C, --nocolor

* Не использовать цветной вывод

-h, --help

* Вывести эту справку и выйти

-V, --version

* Вывести версию программы и выйти

\$Id: qsearch.1,v 1.21 2007/05/11 08:25:06 solar Exp \$

ПЕРЕВОД

- Елена Гаврилова [\(mailto:e.vl.gavrilova@yandex.ru\)](mailto:e.vl.gavrilova@yandex.ru)

Май 2007

qsize

НАЗВАНИЕ

qsize - данные по размеру пакетов

СИНТАКСИС

qsize <опции> <пакет>

ОПИСАНИЕ

Доступные опции: **-[fasSmkbi:vqChV]**

-f, --filesystem

* Показать объем занятого пространства на диске

-a, --all

* Определить размер всех установленных пакетов

-s, --sum

* Включить в вывод сводные данные

-S, --sum-only

* Показать только сводные данные

-m, --megabytes

* Выводить размер в мегабайтах

-k, --kilobytes

* Выводить размер в килобайтах

-b, --bytes

* Выводить размер в байтах

-i, --ignore <аргумент>

* Игнорировать строку шаблона

-v, --verbose

* Предлагать подробный вывод

-q, --quiet

* Предлагать компактный вывод - в частности, не выводить предупреждения

-C, --nocolor

* Не использовать цветной вывод

-h, --help

* Вывести эту справку и выйти

-V, --version

* Вывести версию программы и выйти

\$Id: qsize.1,v 1.22 2007/06/07 18:27:00 solar Exp \$

ПЕРЕВОД

- Елена Гаврилова [\(mailto:e.vl.gavrilova@yandex.ru\)](mailto:e.vl.gavrilova@yandex.ru)

Июнь 2007

qtbz2

НАЗВАНИЕ

qtbz2 - работа с tbz2-пакетами

СИНТАКСИС

qtbz2 <опции> <различные аргументы>

ОПИСАНИЕ

Доступные опции: **-[jstxOvqChV]**

-j, --join

* Упаковать tar.bz2 + xpak в tbz2-архив

-s, --split

* Разбить tbz2-архив на tar.bz2 + xpak

-t, --tarbz2

* Просто разбить tar.bz2-архив

-x, --xpak

* Просто разбить xpak-архив

-O, --stdout

* Писать файлы в стандартный вывод

-v, --verbose

* Предлагать подробный вывод

-q, --quiet

* Предлагать компактный вывод - в частности, не выводить предупреждения

-C, --nocolor

* Не использовать цветной вывод

-h, --help

* Вывести эту справку и выйти

-V, --version

* Вывести версию программы и выйти

\$Id: qtbz2.1,v 1.19 2007/05/11 08:25:06 solar Exp \$

ПЕРЕВОД

- Елена Гаврилова [\(mailto:e.vl.gavrilova@yandex.ru\)](mailto:e.vl.gavrilova@yandex.ru)

Май 2007

quse

НАЗВАНИЕ

quse - узнать, какие пакеты используют USE-флаг

СИНТАКСИС

quse <опции>

ОПИСАНИЕ

Допустимые опции: **-[eavKLDF:NvqChV]**

-e, --exact

* Вместо поиска по шаблону показывать точное соответствие, найденное с помощью функции strcmp

-a, --all

* Выводить все данные IUSE

-K, --keywords

* Использовать KEYWORDS вместо IUSE

-L, --license

* Использовать LICENSE вместо IUSE

-D, --describe

* Описать USE-флаг

-F, --format <аргумент>

* Использовать собственный формат переменных. -F NAME=

-N, --name-only

* Отображать только название пакета

-v, --verbose

* Предлагать подробный вывод

-q, --quiet

* Предлагать компактный вывод - в частности, не выводить предупреждения

-C, --nocolor

* Не использовать цветной вывод

-h, --help

* Вывести эту справку и выйти

-V, --version

* Вывести версию программы и выйти

\$Id: quse.1,v 1.22 2007/05/11 08:25:06 solar Exp \$

ПЕРЕВОД

- Елена Гаврилова [\(mailto:e.vl.gavrilova@yandex.ru\)\](mailto:e.vl.gavrilova@yandex.ru)

Май 2007

qxpak

НАЗВАНИЕ

qxpak - работа с архивами xpak

СИНТАКСИС

qxpak <опции> <различные аргументы>

ОПИСАНИЕ

Доступные опции: **-[lxcd:OvqChV]**

-l, --list

* Выводить содержимое архива

-x, --extract

* Извлечь содержимое архива

-c, --create

* Упаковать каталог/файлы в архив

-d, --dir <аргумент>

* Перейти в указанный каталог

-O, --stdout

* Писать файлы в стандартный вывод

-v, --verbose

* Предлагать подробный вывод

-q, --quiet

* Предлагать компактный вывод - в частности, не выводить предупреждения

-C, --nocolor

* Не использовать цветной вывод

-h, --help

* Вывести эту справку и выйти

-V, --version

* Вывести версию программы и выйти

\$Id: qxpak.1,v 1.19 2007/05/11 08:25:06 solar Exp \$

ПЕРЕВОД

- Елена Гаврилова [\(mailto:e.vl.gavrilova@yandex.ru\)\](mailto:e.vl.gavrilova@yandex.ru)

Справка по gentoolkit

Gentoolkit ([app-portage/gentoolkit](#)) - набор скриптов для администрирования систем, работающих на Gentoo.

- [eclean](#) - утилита для удаления ненужных файлов исходного кода и бинарных пакетов Gentoo
- [enalyze](#) - анализатор установленных пакетов Gentoo
- [epkginfo](#) - отображение метаданных пакетов из портежей
- [equery](#) - утилита для вывода различных данных о пакетах Gentoo
- [eread](#) - Gentoo: утилита для отображения ELOG-файлов из портежей и управления ими
- [eshowkw](#) - Gentoo: утилита для работы с ключевыми словами пакетов
- [euse](#) - Gentoo: редактор USE-флагов из командной строки
- [glsa-check](#) - Gentoo: утилита для локального мониторинга с помощью предупреждений по безопасности GLSA
- [revdep-rebuild](#) - Gentoo: восстановление нарушенных обратных зависимостей

eclean

НАЗВАНИЕ

eclean - утилита для удаления ненужных файлов исходного кода и бинарных пакетов Gentoo

СИНТАКСИС

eclean [*глобальные опции*] ... <*действия*> [*опции действий*] ...

eclean-dist [*глобальные опции, опции файлов исходного кода*] ...

eclean-pkg [*глобальные опции, опции пакетов*] ...

eclean(-dist,-pkg) [*--help, --version*]

ОПИСАНИЕ

eclean - это небольшая утилита, предназначенная для удаления устаревших исходных кодов и бинарных пакетов portage. При регулярном использовании она предотвращает разрастание каталогов DISTDIR и PKGDIR, хотя в отдельных случаях сохранение данных может быть полезно.

По умолчанию eclean препятствует удалению всех файлов исходного кода и бинарных пакетов, соответствующих некоторым билдам, доступным в дереве портежей. Это наиболее безопасный режим работы, поскольку он обеспечивает сохранение тех данных, которые еще могут быть полезны - например, для того, чтобы откатиться на более раннюю версию пакета без повторной загрузки исходного кода или чтобы переустановить пакет, который вы ошибочно удалили, без его перекомпиляции. Но очевидно и то, что в этом случае размер каталогов DISTDIR и PKGDIR будет довольно большим (хотя они не будут увеличиваться бесконечно). Кроме того, в этом режиме утилита медленно работает с файлами исходного кода, так как требует доступа ко всему дереву портежей. Если вы используете опцию *--destructive*, eclean будет сохранять лишь те файлы, которые соответствуют тому или иному пакету, устанавливаемому в данный момент (исходя из точной версии пакета). Эта опция позволяет высвободить гораздо большее пространство; при этом сохраняются файлы исходных кодов для несущественных ревизий и бинарные файлы для переустановки поврежденных пакетов. Но для менее распространенных операций - например, для отката версии или переустановки удаленного пакета - данные будут удалены. Это самый быстрый режим (что очень заметно при обработке файлов исходного кода). Он используется многими сценариями очистки - в качестве примера можно привести yacleaner (по

крайней мере в версии 0.3). Компромиссным вариантом можно считать использование опции `--package-names` в связке с `--destructive` - это предотвратит удаление файлов, соответствующих всем существующим версиям установленных пакетов. В результате при необходимости можно будет легко откатиться на более раннюю версию, не пересобирая пакет и не загружая данные заново, но удалению пакетов это не помешает.

В дополнение к перечисленным основным режимам доступны несколько других опций, позволяющих объявлять особые правила защиты файлов от удаления:

о

опция `--time-limit` удобна для защиты от удаления файлов, которые были созданы ранее заданного времени.

о

опция `--size-limit` (только для файлов исходного кода) удобна для защиты от удаления файлов, размер которых превышает заданный.

о

опция `--fetch-restricted` (только для файлов исходного кода) удобна для защиты от удаления файлов, загруженных вручную. Однако ее использование влечет за собой замедление работы утилиты (по причине, описанной выше: считывание всех данных из дерева портей).

о

Наконец, вы можете включить отдельные пакеты или категории пакетов в список исключаемых (см. ниже раздел **ИСКЛЮЧАЕМЫЕ ФАЙЛЫ**).

ПАРАМЕТРЫ

Глобальные опции

-C, --nocolor

Отключить цветной вывод

-d, --destructive

Сохранить только минимум данных для переустановки

-e, --exclude-file=<путь>

Назначить путь доступа к файлу исключений

<путь>

Абсолютный путь к файлу исключений, который вы хотите использовать. Если эта опция не используется, путями по умолчанию будут соответственно `/etc/eclean/{packages,distfiles}.exclude` (если они существуют). Используйте `/dev/null`, если этот данный файл у вас имеет стандартное расположение, но вы хотите временно его игнорировать.

-i, --interactive

Требовать подтверждения, прежде чем удалить

-n, --package-names Защищать от удаления все версии (только для опции `--destructive`)

-p, --pretend

Не выполнять действий над данными, а лишь показывать, что именно будет удалено

-q, --quiet

Использовать компактный вывод: сообщать только об ошибках

-t, --time-limit=<время>

Не удалять файлы, которые изменились в указанное <время>

<время> - это определенный промежуток времени: 1у обозначает один год, 2w - две недели, и так далее. Допустимые единицы измерения: у (год), т (месяц), w (неделя), d (день), h (час).

-h, --help

Вывести справку по утилите

-v, --verbose

Использовать подробный вывод служебных сообщений в ходе работы утилиты

-V, --version

Вывести информацию о версии

Действия

distfiles

Удалить файлы из каталога /usr/portage/distfiles (или по другому пути, определенному в переменной DISTDIR в вашем файле /etc/make.conf). Это потребуется практически любому пользователю Gentoo, ведь каталог DISTDIR может достигать весьма больших размеров.

eclean-dist представляет собой аlias eclean с действием distfiles, для упрощения вызова из командной строки.

packages

Удалить файлы из каталога /usr/portage/packages (или по другому пути, определенному в переменной PKGDIR в вашем файле /etc/make.conf). Это имеет смысл, в частности, если у вас выставлены флаги FEATURES buildpkg или buildsyspkg.

eclean-pkg представляет собой аlias eclean с действием packages, для упрощения вызова из командной строки.

Опции для действия distfiles

-f, --fetch-restricted предотвращать удаление загруженных вручную файлов (только с опцией --destructive)

-s, --size-limit=<размер>

не удалять файлы исходного кода, размер которых превышает заданный <размер>

<размер> - это размер файла: запись 10M означает 10 мегабайт, 200K - 200 килобайт, и так далее. Допустимые единицы измерения - G (гигабайт), M (мегабайт), K (килобайт) и B (байт).

Опции для действия packages

Отдельных опция для данного действия нет.

ИСКЛЮЧАЕМЫЕ ФАЙЛЫ

Исключаемые файлы представляют собой перечни отдельных пакетов или категорий пакетов, которые вы хотите защитить от удаления. Это имеет смысл, если вам необходимо, например, сохранить собранные системные пакеты. При этом используется следующий синтаксис:

о

Пустые строки и строки, начинающиеся с символа "#" (строки комментариев) игнорируются.

о

Допускается только одна запись в строке.

о

Если строка содержит имя категории, например, sys-apps, ни один пакет из данной категории не будет удален. Запись sys-apps/* также допускается, поскольку она более наглядна, но это НЕ ОЗНАЧАЕТ, что подстановочные символы поддерживаются в каких-либо других целях.

о

Если строка содержит название пакета (например, app-shells/bash), данный пакет удаляться не будет. Указание атомов с версией, например, >=app-shells/bash-3, НЕ ПОДДЕРЖИВАЕТСЯ. Кроме того, обязательно указывать полное имя пакета (с категорией).

о

Если строка содержит имя пакета с восклицательным знаком впереди (например, !sys-apps/portage), данный пакет будет исключен из числа защищенных от удаления. Это имеет смысл только в том случае, если от удаления защищена категория, к которой он относится.

о

При защите файлов исходного кода, строка может содержать и имя файла. Это имеет смысл в том случае, если у вас имеются файлы, не фигурирующие в ебилдах - например, файлы локализации OpenOffice.org i18n (скажем, helpcontent_33_unix.tgz). Другим примером ситуации, в которой следует использовать такой синтаксис, может служить предотвращение удаления данных, не имеющих ебилда в дереве портежей или в любом из подключенных оверлеев.

о

eclean также проинформирует вас об устаревших пакетах, установленных в вашей системе, как если бы она располагала данными о соответствующем им файле/файлах. Если вы хотите защитить от удаления все установленные источники исходного кода, сначала запустите eclean в режиме симуляции. Затем проверьте, для каких источников данных утилите не удалось найти файл/файлы и добавить соответствующие записи в файл distfiles.exclude. Только после этого можно снова вызвать eclean.

По умолчанию, для действия "packages" (или "distfiles") будет использоваться путь /etc/eclean/packages.exclude, если он существует (или, соответственно, distfiles.exclude). Это поведение можно переопределить, используя опцию --exclude-file.

ПРИМЕРЫ

Удалить только исходные коды; по каждому удаляемому архиву будет выдан запрос на подтверждение удаления:

```
# eclean -i distfiles
```

Проверить, какие бинарные пакеты можно удалить; вывод результатов - без цвета:

```
# eclean -Cp packages
```

Удалить бинарные файлы удаленных пакетов, но сохранить все версии установленных:

```
# eclean-pkg -d -n
```

Удалить все исходные коды, за исключением файлов установленных пакетов (точное соответствие версии), которые были созданы менее одного месяца назад, размером 50Мб или имеют ограничения по загрузке:

```
# eclean-dist -d -t1m -s50M -f
```

По расписанию crontab каждое воскресенье в 1.00 автоматически удалять ненужные пакеты в наиболее безопасном режиме, а затем - исходные коды в режиме destructive, но не удалять файлы новее недели:

```
0 1 * * sun \ \ eclean -C -q packages ; eclean -C -q -d -t1w distfiles
```

ПРИМЕЧАНИЕ

При запуске и поиске исходных кодов для удаления eclean будет выдавать сообщения обо всех устаревших пакетах, установленных в вашей системе. Соответствующие источники данных могут быть не защищены от удаления, если переменная SRC_URI не записана в базе данных установленного пакета. В последних версиях portage/pkgcore переменная SRC_URI не записывается.

ОШИБКИ

Решение о том, удалять или нет исходные коды, принимается на основе переменных SRC_URI ебилдов. Это означает, что, когда ебилд обращается к файлам, не указанным в его переменной SRC_URI, eclean, как правило, удаляет такие файлы. Это ошибка ебилда; сообщайте о таких ошибках на <http://bugs.gentoo.org>. В безопасном режиме (используется по умолчанию, с отключенной опцией --destructive) утилита может работать крайне медленно, и это можно поправить только в обход API портежей.

СМ. ТАКЖЕ

Тред форума Gentoo, с которого началась история eclean:

<http://forums.gentoo.org/viewtopic.php?t=3011>

Заявка на багтрекер с просьбой включить eclean в gentoolkit:

http://bugs.gentoo.org/show_bug.cgi?id=33877

АВТОРЫ

- Thomas de Grenier de Latour (tgl) [\[degrenier@easyconnect.fr\]\(mailto:degrenier@easyconnect.fr\)\](mailto:degrenier@easyconnect.fr)
Модули были переписаны:
- Brian Dolbec (dol-sen) [\[brian.dolbec@gmail.com\]\(mailto:brian.dolbec@gmail.com\)\](mailto:brian.dolbec@gmail.com)

ПЕРЕВОД

- Елена Гаврилова [\[e.vl.gavrilova@yandex.ru\]\(mailto:e.vl.gavrilova@yandex.ru\)\](mailto:e.vl.gavrilova@yandex.ru)

0.4.1

ENALYZE

НАЗВАНИЕ

enalyze - анализатор установленных пакетов Gentoo

СИНТАКСИС

enalyze [глобальные опции] модуль [локальные опции] ЦЕЛЬ

ОПИСАНИЕ

enalyze представляет собой набор модулей, предназначенных для анализа состояния установленных пакетов Gentoo, а именно проверки USE-флагов или ключевых слов, с которыми устанавливались данные пакеты, и их текущей маскировки.

По вашему желанию утилита также может генерировать или восстанавливать файлы /etc/portage/package.*.

ОСТОРОЖНО: Данная утилита находится в стадии бета-тестирования, поэтому некоторые возможности и опции в дальнейшем могут быть изменены. Все файлы, создаваемые **enalyze**, сохраняются в вашей домашней директории и без вашего вмешательства не могут никак повлиять на систему.

ГЛОБАЛЬНЫЕ ОПЦИИ

-h, --help

Вывести справочную информацию.

-q, --quiet

Выводить как можно более краткую информацию. С некоторыми модулями использование этой опции может ускорить работу утилиты.

-C, --no-color

Не использовать цветной вывод.

-N, --no-pipe

Отключить автоматическое обнаружение перенаправления. Используйте эту опцию, если вы не хотите, чтобы **analyze** проверял, отображается ли вывод утилиты на экране или передается другой команде, и в зависимости от этого использовал определенные настройки цвета и степени подробности выводимой информации.

-V, --version

Отобразить используемую версию **Gentoolkit**. Если вы оформляете заявку на багтрекере, включайте в нее вывод команды с этой опцией. (см. ниже раздел **ОШИБКИ**)

МОДУЛИ

analyze использует систему модулей. Каждый модуль имеет полное и сокращенное название. В списке ниже используются обозначения вида "**модуль (m)**", где *m* - сокращенное, а **модуль** - полное имя.

Чтобы просмотреть **справку** по интересующему вас модулю, используйте **-h** или **--help** либо как глобальную опцию (указав ее между **analyze** и названием модуля), либо локально (после названия модуля).

analyze (a) [ОПЦИИ] ЦЕЛЬ

Анализирует все установленные пакеты по ЦЕЛи.

ЦЕЛЬ:

use

Анализирует пакеты, установленные с указанными USE-флагами, и выводит результаты.

pkguse

Анализирует информацию о USE-флагах из файла PKGUSE для установленного пакета; этот файл содержит только настройки флагов из каталога /etc/portage/package.use в момент установки.

keywords

Анализирует записанные ключевые слова и выводит результаты.

unmask

Анализирует установленные пакеты и дерево портежей на предмет пакетов, требующих размаскировки, и выводит результат.

ЛОКАЛЬНЫЕ ОПЦИИ:

-u, --unset

Включить в обработку также используемые USE-флаги, которые не были установлены для некоторых пакетов.

-v, --verbose

Вывести более подробный результат, в том числе сведения о ходе выполнения текущего задания.

ПРИМЕРЫ: `analyze a --verbose --unset use`

Будет выведен отчет обо всех USE-флагах, использованных при установке пакетов. (`--unset`) В отчет будут включены все флаги, которые были использованы, но не были установлены для отдельных пакетов. (`--verbose`) Кроме того, будет выведен перечень пакетов, которые использовали настройки USE-флагов. Утилита выведет полный список USE-флагов и сообщит сведения о каждом флаге до 3 раз с указанием его статуса {"+", "-"; " " обозначает неустановленный флаг} перед именем флага. Вывод будет цветным: красным цветом обозначается отключенный флаг, синим - включенный, обычным текстом - неустановленный.

rebuild (r) [ОПЦИИ] ЦЕЛЬ

Создать список всех пакетов для настроек *ЦЕЛи*, которые необходимы для настроек, отличных от стандартных.

ЦЕЛЬ:

use

Анализирует USE-флаги и выводит результат.

keywords

Анализирует ключевые слова и выводит результат.

unmask

Анализирует установленные пакеты и дерево портежей на предмет пакетов, которые требуют размаскировки, и выводит результат / создает новый файл в каталоге /etc/portage/package.unmask.

ЛОКАЛЬНЫЕ ОПЦИИ:

-a, --all

Создать файлы/вывод для всех *ЦЕЛей*, для которых это необходимо. (Эта функция еще не реализована.)

-e, --exact

Использовать в записях префикс пакета =, а также информацию о версии.

Образец: =КАТЕГОРИЯ/ПАКЕТ-ВЕРСИЯ флаг1 флаг2

-p, --pretend

Не перенаправлять вывод в файл, а отображать его на экране.

-v, --verbose

Вывести более подробный результат, в том числе сведения о ходе выполнения текущего задания.

ПРИМЕРЫ:

analyze rebuild -p use

Будет проанализирована база данных установленных пакетов и текущие настройки системных USE-флагов, и утилита выведет результаты в следующем виде:

КАТЕГОРИЯ/ПАКЕТ -флаг1 -флаг2 флаг3 флаг4...

clean (c) [ОПЦИИ] ЦЕЛЬ

Очищает все пакеты от настроек, указанных в аргументе *ЦЕЛЬ*, которые по сравнению с текущими и ебилдом пакета являются устаревшими. (Эта функция еще не реализована.)

ЦЕЛЬ:

use

Анализирует USE-флаги и файл(ы) в каталоге /etc/portage/package.use на предмет повторяющихся или более не используемых пакетом записей.

keywords

Анализирует ключевые слова и файл(ы) в каталоге /etc/portage/package.keywords, которые более не нужны.

unmask

Анализирует установленные пакеты, файл(ы) в каталоге /etc/portage/package.unmask и дерево портежей на предмет пакетов, которые более не требуют размаскировки.

ЛОКАЛЬНЫЕ ОПЦИИ:

-a, --all

Очистить файлы/вывод для всех ЦЕЛей, для которых это необходимо.(Эта функция еще не реализована.)

-p, --pretend

Вывод не будет перенаправлен в файл, а отобразится на экране терминала.

-v, --verbose

Вывести более подробный результат, в том числе сведения о ходе выполнения текущего задания.

ОШИБКИ

Об обнаруженных ошибках сообщайте на <http://bugs.gentoo.org>.

АВТОРЫ

- Brian Dolbec [\(mailto:brian.dolbec@gmail.com\)](mailto:brian.dolbec@gmail.com), 2010

ПЕРЕВОД

- Елена Гаврилова [\(mailto:e.vl.gavrilova@yandex.ru\)](mailto:e.vl.gavrilova@yandex.ru)

Февраль 2010

EPKGINFO

НАЗВАНИЕ

epkginfo - отображение метаданных пакетов из портежей

СИНТАКСИС

epkginfo [опции] пакет или атом

ОПИСАНИЕ

Epkginfo - это системный алиас команды **equery meta**. Полный список доступных опций и дополнительные примеры см. в man-руководстве по **equery**.

ПРИМЕРЫ

```
* app-portage/gentoolkit [portage]
Location:      /usr/portage/app-portage/gentoolkit
Herd:          tools-portage (tools-portage@gentoo.org)
Maintainer:    None specified
Upstream:      None specified
Keywords:      0.2.4.5: alpha amd64 arm hppa ia64 m68k ~mips ppc ppc64
                  s390 sh sparc x86 ~x86-fbsd
Keywords:      0.3.0_rc5: ~alpha ~amd64 ~arm ~hppa ~ia64 ~m68k ~mips
                  ~ppc ~ppc64 ~s390 ~sh ~sparc ~x86 ~sparc-fbsd
                  ~x86-fbsd
Keywords:      0.3.0_rc6: ~alpha ~amd64 ~arm ~hppa ~ia64 ~m68k ~mips
                  ~ppc ~ppc64 ~s390 ~sh ~sparc ~x86 ~sparc-fbsd
                  ~x86-fbsd
Keywords:      0.3.0_rc7: ~alpha ~amd64 ~arm ~hppa ~ia64 ~m68k ~mips
                  ~ppc ~ppc64 ~s390 ~sh ~sparc ~x86 ~sparc-fbsd
                  ~x86-fbsd
```

АВТОРЫ

Douglas Anderson [\(mailto:douglasjanderson@gmail.com\)](mailto:douglasjanderson@gmail.com)

ОШИБКИ

Об обнаруженных ошибках сообщайте на <http://bugs.gentoo.org>

ПЕРЕВОД

- Елена Гаврилова [\(mailto:e.vl.gavrilova@yandex.ru\)\](mailto:e.vl.gavrilova@yandex.ru)

Август 2009

EQUERY

НАЗВАНИЕ

equery - вывод различных данных о пакетах Gentoo

СИНТАКСИС

equery [глобальные опции] **module** [локальные опции]

ОПИСАНИЕ

Equery представляет собой набор модулей для вывода актуальной информации о пакетах, файлах и USE-флагах Gentoo.

ГЛОБАЛЬНЫЕ ОПЦИИ

-h, --help

Выводить справочное сообщение.

-q, --quiet

По возможности уменьшить объем выводимой информации. Для некоторых модулей использование этой опции может ускорить вывод.

-C, --no-color

Не использовать цветной вывод.

-N, --no-pipe

Отключить автоматическое обнаружение каналов. Используйте эту опцию, если не хотите, чтобы **equery** выяснял, идет ли вывод на экран или перенаправляется другой программе, и соответственно изменял настройки цвета и степени подробности выводимой информации.

-V, --version

Вывести версию **Gentoolkit**. Пожалуйста, включайте вывод equery с данной опцией в сообщения об ошибках (см. ниже раздел **ОШИБКИ**).

МОДУЛИ

Equery использует модульную систему. Каждый модуль имеет как полное, так и сокращенное имя. В приведенном ниже списке используются обозначения вида "**module (m)**", где *m* - сокращенное имя, а *module*, соответственно - полное.

Вы можете просмотреть **справку** по интересующему вас модулю, используя опцию **-h, --help** либо глобально (между **equery** и именем модуля) или локально(после имени модуля).

belongs (b) [ОПЦИИ] ФАЙЛ

Выводит пакет, которому принадлежит **ФАЙЛ**.

Примечание: как правило, **ФАЙЛ** принадлежит только одному пакету. Если один и тот же файл принадлежит нескольким пакетам одновременно, сообщите об этом разработчикам (см. ниже раздел **ОШИБКИ**).

ЛОКАЛЬНЫЕ ОПЦИИ:

-f, --full-regex

Запрос представляет собой регулярное выражение.

-e, --early-out

Завершить работу после первого найденного соответствия. Как правило, это безопасно и полезно при поиске пакета-владельца одного файла.

-n, --name-only

Не выводить версию.

ПРИМЕРЫ: equery belongs --early-out /usr/bin/euse

Выяснить, какой пакет установил команду. emerge -p \$(equery -q belongs -nf '^/usr/bin/g?vim.*')

Заставить **emerge** переустановить или обновить все пакеты, которые установили файл, совпадающий с шаблоном.

changes (c) [ОПЦИИ] ПАКЕТ

Отобразить запись Gentoo ChangeLog для последней доступной для установки версии **ПАКЕТА**.

ЛОКАЛЬНЫЕ ОПЦИИ:

-l, --latest

Вывести только последнюю запись чейнджлога. Зачастую, если не было мажорного обновления версии, изменения фиксируются в чейнджлоге без заголовка версии; используйте эту опцию для вывода подобных записей.

-f, --full

Вывести полный ChangeLog.

Совет: Используйте конвейер () для постраничного вывода, например, с помощью 'less'.

--limit=ЧИСЛО

Ограничить ЧИСЛО выводимых записей. Используйте эту опцию в связке с **--full**. **--limit=3** выведет три последних записи.

--from=ВЕРСИЯ

Установить ВЕРСИЮ, начиная с которой отображать. Самостоятельное использование этой опции аналогично передаче модулю **changes** атому пакета на заданном интервале, например, '>=foo/bar-1.5'. Опцию можно использовать вместе с **--to**, если требуется задать более сложные условия поиска.

--to=VER

Установить ВЕРСИЮ, до которой отображать. (См. **--from**)

ПРИМЕРЫ:

equery changes portage

Выводить запись в чейнджлоге Gentoo для последней доступной для установки версии Portage. equery changes '=sys-apps/portage-2.1.6*'

Использовать синтаксис атома Portage. (Для вывода справки выполните **man 5 ebuild**.) equery changes portage --from=2.2_rc1 --to=2.2

Выводить любые записи чейнджлога на указанном интервале версий.

check (k) [ОПЦИИ] ПАКЕТ

Проверяет отметки времени и контрольные суммы файлов, принадлежащих ПАКЕТу; ПАКЕТ здесь – один из установленных пакетов.

ЛОКАЛЬНЫЕ ОПЦИИ:

-f, --full-regex

Запрашивать по регулярному выражению.

-o, --only-failures

Отображать только те пакеты, которые не прошли проверку.

ПРИМЕРЫ: equery check --only-failures '*'

Проверить отметки времени и контрольные суммы всех установленных пакетов и вывести только те пакеты, которые не прошли эту проверку. equery check 'dev-python/*' dev-lang/python

Проверить каждый установленный пакет из категории **dev-python** и сам Python.

depends (d) [ОПЦИИ] ПАКЕТ

Вывести список всех зависимостей *ПАКЕТА*.

ЛОКАЛЬНЫЕ ОПЦИИ:

-a, --all-packages

Включить в запрос неустановленные зависимости. Операция может занять некоторое время.

-D, --indirect

Выявлять как явные, так и неявные зависимости.

--depth=ЧИСЛО

Ограничить глубину неявной зависимости ЧИСЛОм. Использование значения **--depth=0** эквивалентно неиспользованию **--indirect**.

ПРИМЕРЫ: equery depends --indirect xulrunner

Выяснить, какие пакеты "притянули" интересующий вас пакет в систему.

depgraph (g) [ОПЦИИ] ПАКЕТ

Выводит схему явных зависимостей для каждой версии *ПАКЕТА* по запросу. Схема зависимостей представляет собой дерево пакетов, показывающее отношения между пакетами и их зависимостями.

ЛОКАЛЬНЫЕ ОПЦИИ:

-A, --no-atom

Не выводить атом зависимости, соответствующий пакету.

-U, --no-useflags

Не отображать USE-флаги.

-l, --linear

Отображать схему зависимостей в плоском виде, без отступов. При выборе этой опции глубина рекурсии будет показана в квадратных скобках перед именем пакета. Такой режим просмотра удобен на нешироком экране терминала.

--depth=ЧИСЛО

Ограничить глубину отображаемых зависимостей ЧИСЛОм. Запись **--depth=0** обозначает, что максимальная глубина не указана. Значение по умолчанию - 1.

ПРИМЕРЫ: equery depgraph --depth=0 portage

Просмотреть полное дерево явных и неявных компиляционных, рабочих и постустановочных зависимостей пакета.

files (f) [ОПЦИИ] ПАКЕТ

Выводит файлы и каталоги, устанавливаемые *ПАКЕТОМ*.

ЛОКАЛЬНЫЕ ОПЦИИ:

-m, --md5sum

Включить в вывод контрольную сумму файла.

-s, --timestamp

Включить в вывод отметку времени.

-t, --type

Включить в вывод тип файла.

--tree

Отображать файлы как дерево. Эта опция отключает все прочие локальные опции.

-f, --filter=ПРАВИЛА

Фильтровать вывод по типу файла.

ПРАВИЛА

Представляют собой список, через запятую, расширений файлов (без пробелов); вы можете выбирать из:
dir, obj, sym, dev, path, conf, cmd, doc, man, info

ПРИМЕРЫ: equery files --tree vlc

Просмотреть полное дерево файлов, установленных пакетом. equery files --filter=cmd vlc

Выяснить, куда при установке были помещены исполняемые файлы пакета.

has (a) [ОПЦИИ] КЛЮЧ ЗНАЧЕНИЕ

Выводит все установленные пакеты, отвечающие заданным атрибутам *ПАКЕТА*.

Примечание: **КЛЮЧ** чувствителен к регистру. Кроме того, опция **has** в настоящее время не умеет сопоставлять значения в зависимости от типа запрашиваемой информации: она лишь выполняет поиск по строке и сообщает, какие пакеты имеют заданное аргументом в командной строке **ЗНАЧЕНИЕ**. Это запрос информации общего характера, выполняемый с помощью функции portage dbapi.aux_get().

Внимание: качество результата, который вы увидите, зависит от качества поиска (с учетом ограничений метода сравнения) и записанных данных, доступных в vardb. (См. раздел *ПРИМЕРЫ*.)

ЛОКАЛЬНЫЕ ОПЦИИ:**-I, --exclude-installed**

Исключить из вывода установленные пакеты.

-o, --overlay-tree

Включить в обработку пакеты из оверлеев.

-p, --portage-tree

Включить в обработку все пакеты из дерева портежей. Используйте эту опцию для поиска среди всех стандартных пакетов Gentoo, включая неустановленные.

-F, --format=ШАБЛОН

Изменить формат вывода по умолчанию для отвечающих запросов пакету с помощью строки **ШАБЛОНА**. См. ниже в разделе об опции **-format** для модуля **list** описание этого аргумента.

ВЫВОД:

См. ниже раздел **ВЫВОД** для модуля **list**.

ПРИМЕРЫ:

equery has SLOT 2.4

Просмотреть все установленные пакеты Gentoo с SLOT = "2.4".

equery has repository sunrise

Просмотреть все установленные пакеты Gentoo, которые были записаны для установки из eбилдов из оверлея sunrise.

equery has EAPI 2

Просмотреть все установленные пакеты Gentoo, которые были установлены из eбилдов с EAPI 2.

hasuse (h) [ОПЦИИ] USE-ФЛАГ

Выводит все установленные пакеты, имеющие данный USE-флаг.

Примечание: на данный момент **hasuse** не отображает информацию о том, собраны ли пакеты с данным флагом, а только показывает, для каких пакетов флаг доступен. (См. *ПРИМЕРЫ*)

ЛОКАЛЬНЫЕ ОПЦИИ:

-I, --exclude-installed

Исключить из вывода установленные пакеты.

-o, --overlay-tree

Включить в обработку пакеты из оверлеев.

-p, --portage-tree

Включить в обработку все пакеты из дерева портежей. Используйте эту опцию для поиска среди всех стандартных пакетов Gentoo, включая неустановленные.

-F, --format=ШАБЛОН

Изменить формат вывода по умолчанию для отвечающих запросов пакету с помощью строки *ШАБЛОНа*. См. ниже в разделе об опции **--format** для модуля **list** описание этого аргумента.

ВЫВОД:

(См. ниже раздел *ВЫВОД* для модуля **list**)

ПРИМЕРЫ:

equery hasuse -pI perl

Просмотреть все пакеты Gentoo с USE-флагом "perl", за исключением уже установленных.

USE="perl"; for PKG in \$(equery -q hasuse \$USE); do echo \$PKG: \$(equery -q uses \$PKG |grep \$USE); done

Этот однострочный сценарий для Bash вызывает **hasuse** для составления перечня пакетов с определенным USE-флагом и **uses**, чтобы проверить, включен или отключен флаг. Для изменения запроса редактируйте значение **USE="perl"**.

list (l) [ОПЦИИ] ПАКЕТ

Выводит установленные версии *ПАКЕТА* или всех пакетов, удовлетворяющих шаблону поиска.

ЛОКАЛЬНЫЕ ОПЦИИ:

-d, --duplicates

Выводить только пакеты, для которых установлены несколько версий.

-f, --full-regex

Искать по регулярному выражению.

-m, --mask-reason

Сообщать причину, по которой тот или иной пакет замаскирован.

-I, --exclude-installed

Исключить из вывода установленные пакеты.

-o, --overlay-tree

Включить в обработку пакеты из оверлеев.

-p, --portage-tree

Включить в обработку все пакеты из дерева портежей. Используйте эту опцию для поиска среди всех стандартных пакетов Gentoo, включая неустановленные.

-F, --format=TMPL

Изменить формат вывода по умолчанию для отвечающих запросов пакету с помощью строки **ШАБЛОНа**. **ШАБЛОН** может содержать следующие подстановочные поля:

\$cp - Только категория и название пакета (например, 'app-portage/gentoolkit').

\$cpv - Категория, название пакета и его полная версия (например, 'app-portage/gentoolkit-0.3.0_rc10-r1').

\$category - Только категория (например, 'app-portage').

\$name - Только название пакета (например, 'gentoolkit').

\$version - Версия пакета (без указания ревизии) (например, '0.3.0_rc10').

\$revision - Ревизия пакета (например, 'r1').

\$fullversion - Версия пакета с ревизией (например, '0.3.0_rc10-r1').

\$slot - Слот пакета.

\$repo - Имя репозитария пакета (например, 'gentoo').

\$mask - Поле для указания маски пакета (~M-??), подробнее см. ниже в разделе **ВЫВОД**.

\$mask2 - Подробное описание статуса маскировки пакетов.

\$location - Поле для указания расположения пакета (**IPO-**), подробнее см. ниже в разделе **ВЫВОД**.

Кроме перечисленных подстановочных полей, строка шаблона может содержать любой произвольный текст. Аналогично переменным интерпретатора bash, здесь можно использовать фигурные скобки для отличия имен переменных от объемлющего текста.

ВЫВОД:

```
$ equery list binutils * Searching for binutils ... [I--] [??] sys-
devel/binutils-2.18-r1:i686-pc-linux-gnu-2.18 [IP-] [ ~] sys-
devel/binutils-2.19.1-r1:i686-pc-linux-gnu-2.19.1
```

Поле расположения (**[IPO-]**):

Первое поле - расположение и статус установки пакета. Оно содержит три символа в квадратных скобках. **I** означает, что на данный момент пакет установлен. **P** означает, что пакет доступен в дереве портежей. **O** означает, что пакет доступен по крайней мере в одном оверлее. **-** не означает ничего, занимая пустое поле. Так, если вы видите **[I-O]**, это значит, что пакет установлен и доступен в оверлее, но в дереве портежей его нет.

Поле маски (**[~M-??]**):

Второе поле - статус маскировки пакета. Пустые скобки означают, что пакет не замаскирован. Тильда ~ означает, что пакет замаскирован по ключевому слову: например, ваша система из стабильной ветки, а пакет помечен как тестируемый. **M** означает жесткую маскировку: например, мейнтайнер пакета посчитал, что пакет не пригоден для широкого использования. **-** означает маскировку по архитектуре: например, у вас amd64, а пакет работает только на архитектуре x86. В последних версиях обозначение **??** появляется лишь в том случае, если в поле расположения стоит **[I--]**. В связке друг с другом они означают, что пакет был установлен из дерева портежей или из оверлея, но затем были удалены из хранилища; в результате утилиты **equery** не может определить статус маскировки пакета.

Название пакета:

Третье поле - полное имя пакета с версией.

Слот:

Четвертое поле, после двоеточия - слот пакета. По умолчанию это **0**. Для обнаружения всех пакетов с несколькими установленными слотами используйте опцию **--duplicates**.

Примечание: Для определения расположения, статуса маскировки и слота пакета требуется дополнительное время; если вам не нужна развернутая информация, вы можете глобально передать утилите **equery** опцию **--quiet**, чтобы ускорить обработку.

Примечание: Если при использовании опции **--quiet** не было найдено соответствий запросу, модуль ***equery list*** не сообщает об ошибке, а завершает работу и возвращает 3.

ПРИМЕРЫ: **equery list '*'**

Выводить все установленные пакеты. В версиях **Gentoolkit** ниже 0.3.0 этот запрос эквивалентен '**equery list'**.

equery list -op mozilla-firefox

Выводить все доступные версии пакета, точно соответствующие 'mozilla-firefox'. В версиях **Gentoolkit** ниже 0.3.0 этот запрос эквивалентен '**equery list --exact-name -o -p mozilla-firefox**'. **equery list '*zilla*'**

Выводить все пакеты, которые содержат 'zilla' (нечеткий поиск). В версиях **Gentoolkit** ниже 0.3.0 этот запрос эквивалентен '**equery list zilla**'. **equery list 'www-client/*'**

Выводить все пакеты из категории **www-client**. В версиях **Gentoolkit** ниже 0.3.0 этот запрос эквивалентен '**equery list --category=www-client**'. **equery list --duplicates '*'**

Выводить все пакеты с несколькими установленными версиями. В версиях **Gentoolkit** ниже 0.3.0 этот запрос эквивалентен '**equery list --duplicates**'. **equery list -F '\$cp:\$slot' '*'**

Получить список атомов слотов для всех установленных пакетов. **equery list -po -F ['\$location'] [\$mask] \$cpv:\$slot ['\$repo'] '*'**

Выводит все пакеты в формате по умолчанию (подробно), а также имя репозитария, который их предоставляет.

meta (m) [ОПЦИИ] ПАКЕТ

Отображает метаданные о *ПАКЕТе*.

meta считывает файл *metadata.xml*, который должен сопровождать все пакеты из дерева портежей. **meta** не считывает ебилды и потому может возвращать только мета-данные, не зависящие от версии.

Поскольку до недавнего времени просматривать *metadata.xml* было довольно затруднительно, а также потому, что мейнтейнеры пакетов должны заполнять лишь небольшую часть файла, для очень многих пакетов подробные мета-данные до сих пор отсутствуют. Подробнее о *metadata.xml* см.:

<http://www.gentoo.org/proj/en/devrel/handbook/handbook.xml?part=2&chap=4>

ЛОКАЛЬНЫЕ ОПЦИИ:

-d, --description

Выводить развернутое описание пакета.

-H, --herd

Отображать группу/-ы сопровождения для пакета. Если не используются каналы и **--quiet** не передается как глобальная опция, будет показан также адрес группы (по умолчанию).

-k, --keywords

Показывать ключевые слова для всех удовлетворяющих запросу версий. **keywords** не выводит все ключевые слова для всех версий, а фильтрует список, чтобы легче было выявить версии, которые должны быть заменены новыми или могут быть удалены из дерева. Фильтрование производится по слоту. Например:

Keywords: 1.35.0-r3:**0**:

Keywords: 1.35.0-r5:**0**: amd64 hppa ppc x86 ~alpha ~arm ~ia64 ~mips ~ppc64 ~s390 ~sh ~sparc

В приведенном выводе **equery meta boost -r5** является последней доступной версией в слоте 0, поэтому перечислены все ключевые слова. Для -r3 действительны ключевые слова "`~amd64 ~hppa ~ppc ~x86`", но, поскольку более высокая версия в том же слоте помечена теми же ключевыми словами или как более стабильная, они не будут показаны. Вместе с тем ключевые слова для маскировки архитектуры (-*) отображаются всегда.

-m, --maintainer

Показать электронный адрес мейнтейнера/-ов пакета. Если доступны мета-данные, будет также отображено имя мейнтейнера и/или описание его работы (по умолчанию).

-u, --useflags

Выводить описания USE-флагов каждого пакета. Такие описания иногда добавляются в файл `metadata.xml` - в случае если флаг необычным образом изменяет пакет или если он слишком редок, чтобы фигурировать в глобальном файле описаний. Теперь **equery uses** умеет выводить и локальные описания, так что эта опция по-прежнему доступна в **meta** только для полноты.

-U, --upstream

Вывести информацию о ключевом разработчике пакета, включая его электронную почту, багтрекер и документацию. На момент написания данного руководства большинство мейнтейнеров не предоставляли такой информации (по умолчанию).

-x, --xml

Вывести обычный XML-файл на экран.

ПРИМЕРЫ:

`equery meta gnucash`

Показать общие сведения о поддержке, в том числе о группе сопровождения, мейнтейнерах и ключевых разработчиках. `equery meta --description screen`

Выяснить, предоставляет ли мейнтейнер пакета развернутое описание. `equery -N meta -H gnome |grep -o --color=never '[^(\]*@gentoo.org'`

Извлечь электронный адрес группы сопровождения (может быть, вы хотите отправить им письмо с благодарностью). Не забывайте о том, что сообщения об ошибках следует направлять не по этому адресу, а размещать на сайте `bugs.gentoo.org`. В приведенном примере будет извлечен один или (если есть) несколько адресов; если установлено значение **no-herd**, утилита не возвратит ничего.

size (s) [ОПЦИИ] ПАКЕТ

Выводит суммарный размер файлов, составляющих запрошенный ПАКЕТ.

ЛОКАЛЬНЫЕ ОПЦИИ:

-b, --bytes

Вывести размер пакета в байтах.

-f, --full-regex

Аргументом запроса является регулярное выражение.

ПРИМЕРЫ: `equery -q size 'www-client/*'`

Получить односторонний отчет о количестве файлов и их суммарном размере (в байтах), для каждого установленного пакета в категории.

uses (u) [ОПЦИИ] ПАКЕТ

Позволяет просмотреть состояние и описания USE-флагов для запрошенного ПАКЕТА.

ЛОКАЛЬНЫЕ ОПЦИИ:

-a, --all

Вывести все версии пакета. Если эта опция не используется, **equery** выведет лучшую доступную версию.

ПРИМЕРЫ:

`equery uses app-misc/beagle`

Выяснить, какие USE-флаги включены для пакета.

```
USE="perl"; for PKG in $(equery -q hasuse $USE); do echo $PKG: $(equery -q uses $PKG |grep $USE); done
```

Эта односторонковая команда Bash вызывает **hasuse**, чтобы вывести список пакетов с заданным USE-флагом, и **uses**, чтобы проверить, включен флаг или выключен. Для изменения запроса используйте другое значение `USE="perl"`.

which (w) [ОПЦИИ] ПАКЕТ

Выводит путь к ебилду, который будет использован Portage с текущими настройками.

ЛОКАЛЬНЫЕ ОПЦИИ:

-m, --include-masked

Просмотреть путь к последней опубликованной версии ебилда.

ПРИМЕРЫ:

`less $(equery which xorg-server)`

Найти последний опубликованный ебилд из доступных для установки.

ОШИБКИ

Об обнаруженных ошибках сообщайте на <http://bugs.gentoo.org>.

АВТОРЫ

- Karl Trygve Kalleberg [\(mailto:karltk@gentoo.org\)](mailto:karltk@gentoo.org), 2003
- Katerina Barone-Adesi [\(mailto:katerinab@gmail.com\)](mailto:katerinab@gmail.com), 2004
- Douglas Anderson [\(mailto:douglasjanderson@gmail.com\)](mailto:douglasjanderson@gmail.com), 2009

ПЕРЕВОД

- Елена Гаврилова [\(mailto:e.vl.gavrilova@yandex.ru\)](mailto:e.vl.gavrilova@yandex.ru)

Август 2009

eread

НАЗВАНИЕ

`eread` - - Gentoo: утилита для отображения ELOG-файлов из портежей и управления ими

СИНТАКСИС

`eread`

ОПИСАНИЕ

Эта утилита предназначена для вывода в терминале содержимого ELOG-файлов, генерируемых Portage версии 2.1 и выше, и работы с ними.

ПЕРЕМЕННЫЕ ОКРУЖЕНИЯ

Для отображения ELOG-файлов утилита `eread` использует переменную окружения `PAGER`. Если эта переменная не установлена, будет использовано значение по умолчанию, `/usr/bin/less`.

ПЕРЕВОД

- Елена Гаврилова [\[e.vl.gavrilova@yandex.ru\]\(mailto:e.vl.gavrilova@yandex.ru\)\](mailto:e.vl.gavrilova@yandex.ru)

1.0

eshowkw

НАЗВАНИЕ

eshowkw - Gentoo: утилита для работы с ключевыми словами пакетов

СИНТАКСИС

eshowkw [ОПЦИИ] [--] [пакет [пакет ...]]

ОПИСАНИЕ

Выводит ключевые слова для пакета (-ов), указанного (-ых) в аргументе или для пакета, расположенного в текущем рабочем каталоге.

ОПЦИИ

package

Проверяемые пакеты. (По умолчанию будут обрабатываться обрабатываться билды в текущем каталоге.)

-h | --help

Вывести информацию о синтаксисе утилиты.

-v | --version

Вывести версию утилиты и выйти.

-a=АРХИТЕКТУРА[,АРХИТЕКТУРА ...] | --arch=АРХИТЕКТУРА[,АРХИТЕКТУРА ...]

Фильтровать по указанной архитектуре, исключая остальные.

-A {top,bottom} | --align {top,bottom}

Указать тип выравнивания описаний. (По умолчанию - по нижнему краю (bottom).)

-T {archlist,versionlist} | --top-position {archlist,versionlist}

Указать поля, которые должны присутствовать в верхнем листинге. (По умолчанию - archlist.)

-B | --bold

Отображать колонки через одну жирным шрифтом, для удобства восприятия. (По умолчанию - нет (False).)

-C | --color

Принудительно использовать цветной вывод (По умолчанию - нет (False).)

-O | --overlays

Искать в оверлеях. (По умолчанию - нет (False).)

-P | --prefix

Включать в вывод префиксные ключевые слова. (По умолчанию - нет (False).)

-S | --ignore-slot

При выявлении избыточных пакетов не принимать во внимание слоты. (По умолчанию - нет (False).)

ОШИБКИ

Об обнаруженных ошибках сообщайте на [\[http://bugs.gentoo.org\]](http://bugs.gentoo.org)(<http://bugs.gentoo.org>>).

ПЕРЕВОД

- Елена Гаврилова [\[e.vl.gavrilova@yandex.ru\]](mailto:e.vl.gavrilova@yandex.ru)(<mailto:e.vl.gavrilova@yandex.ru>)

EUSE

НАЗВАНИЕ

euse - Gentoo: редактор USE-флагов из командной строки

СИНТАКСИС

euse <опция> [подопция] [флаги]

ОПИСАНИЕ

Утилита *euse* предназначена для настройки (включения/отключения) USE-флагов в /etc/make.conf без прямого редактирования этого файла. Она используется также для получения подробных сведений о USE-флагах: описание, состояние (включен/отключен), тип флага (глобальный/локальный) и т.д.

ОПЦИИ

-E, --enable

Включить USE-флаг(и) в файле make.conf. В качестве параметров принимает один или более USE-флагов, через пробел.

-D, --disable

Отключить USE-флаг(и) в файле make.conf. Помещает символ '-' перед именем флага и добавляет его к значению USE, указанному в make.conf. В качестве параметров принимает один или более USE-флагов, через пробел.

-P, --prune

Удаляет USE-флаг(и) из файла make.conf. Удаляет все положительные и отрицательные ссылки на заданные USE-флаги из make.conf.

-i, --info

Выводит подробную информацию о USE-флаге/-ах. Если аргументов не задано, будут отображены сведения по всем флагам. Если заданы один или более аргументов (через пробел), выводимая информация будет относится только к данным флагам.

-I, --info-installed

Аналогично --info за исключением того, что будут отображены и установленные на момент выполнения команды пакеты, использующие данный флаг.

Формат вывода таков:

[- c]alpha - indicates that architecture ...

[-]moznocompose (net-www/mozilla):

Disable building of mozilla's web page composer

Расшифровка индикаторов в первом столбце:

is_active

+, если portage считает флаг активным, иначе -

is_in_env

E, если флаг включен в окружении, e если флаг отключен в окружении, ничего, если окружение не определяет флаг.

is_in_make_conf

C, если флаг включен в make.conf, c, если флаг отключен в make.conf, ничего, если make.conf не определяет флаг.

is_in_make_defaults

D, если флаг включен в make.defaults, d, если флаг отключен в make.defaults, ничего, если make.defaults не определяет флаг.

is_in_make_globals

G, если флаг включен в make.globals, g, если флаг отключен в make.globals, ничего, если make.globals не определяет флаг.

За литерой следует имя флага, для локальных флагов - название пакета и, наконец - описания (для локальных флагов - на новой строке).

-a, --active

Отображает все активные USE-флаги и где они активируются (см. описание --info).

-h, --help

Отображает страницу справки, в которой перечислены все доступные ключи с их кратким описанием.

-v, --version

Отображает информацию о версии.

ФАЙЛЫ

/etc/make.conf

/etc/make.profile/make.defaults

/etc/make.globals

\$PORTDIR/profiles/use.desc

\$PORTDIR/profiles/use.local.desc

АВТОР

- Arun Bhanu [codebear@gentoo.org] (<mailto:codebear@gentoo.org>) (оригинальная версия)
- Marius Mauch [genone@gentoo.org] (<mailto:genone@gentoo.org>) (обновление для новой версии euse)
- Jared Hancock (в значительной степени переписана поддержка package.use)

ОШИБКИ

В настоящее время euse не обрабатывает USE-флаги, которые были включены или отключены через use.defaults, use.mask или package.use. Он не вполне корректно распознает и флаг -*.

СМ. ТАКЖЕ

ufed(8),

Сценарий /usr/bin/euse.

ПЕРЕВОД

- Елена Гаврилова [e.vl.gavrilova@yandex.ru] (<mailto:e.vl.gavrilova@yandex.ru>)

2004-10-17

glsa-check

НАЗВАНИЕ

glsa-check - Gentoo: утилита для локального мониторинга с помощью предупреждений по безопасности GLSA

СИНТАКСИС

glsa-check <_опция_> [список-glsa]

Аргумент [список *glsa*] может содержать произвольное количество номеров GLSA, файлов, содержащих GLSA, или специальных идентификаторов: 'all' (все) и 'affected' (выявленные у вас уязвимости).

ОПИСАНИЕ

Данная утилита предназначена для локального мониторинга системы с помощью предупреждений по безопасности Gentoo Linux (GLSA). Прежде чем обращаться на багтрекер, ознакомьтесь со следующей документацией:

<http://www.gentoo.org/security>.

Примечание: для эффективной работы утилиты следует регулярно обновлять локальное дерево портежей.

ОПЦИИ

-l, --list

вывести краткий отчет по GLSA из списка *glsa*, в том числе информацию о наличии известных уязвимостей

-d, --dump, --print

вывести полную информацию по GLSAs из списка *glsa*

-t, --test

проверить, влияют ли GLSA из списка *glsa* на систему, и вывести их номера

-p, --pretend

проверить, относятся ли GLSA из списка *glsa* к вашей системе

-f, --fix

попытаться автоматически наложить заданные исправления GLSA из списка *glsa-list* на вашу систему с помощью emerge. При этом пакеты будут обновлены до последней версии, но если нет пути обновления, пакеты удалятся не будут (экспериментальная опция).

-i, --inject

Поместить данные GLSA в файл *glsa_injected*.

-n, --nocolor

Отключить цветной вывод (опционально).

-h, --help

Вывести эту справку.

-V, --version

Вывести информацию об утилите.

-v, --verbose

Использовать более подробный вывод (опционально).

-c, --cve

Показать номера CVE в режиме списка (опционально).

-q, --quiet

Использовать краткий вывод и не отправлять пустых писем (опционально).

-m, --mail

Отправить письмо с данными GLSA администратору.

ФАЙЛЫ

/var/lib/portage/glsa_injected

Содержит список номеров GLSA, которые были исправлены и никогда не будут отображаться как 'affected' на данной системе. Каждая строка файла должна содержать только один номер GLSA (например, '200804-02').

ПЕРЕВОД

- Елена Гаврилова [\[e.vl.gavrilova@yandex.ru\]](mailto:e.vl.gavrilova@yandex.ru)(mailto:e.vl.gavrilova@yandex.ru)\

0.3

revdep-rebuild

НАЗВАНИЕ

revdep-rebuild - Gentoo: восстановление нарушенных обратных зависимостей

СИНТАКСИС

revdep-rebuild [ОПЦИИ] [--] [ОПЦИИ EMERGE]

ОПИСАНИЕ

Утилита *revdep-rebuild* проверяет библиотеки и бинарные файлы на предмет неудовлетворенных зависимостей совместно используемых библиотек и пытается исправить нарушенные зависимости путем переустановки этих бинарных файлов и библиотек. Ее полезно использовать в тех случаях, когда установка одного пакета нарушает нормальную работу других, зависящих от него.

ОПЦИИ

-C | --nocolor

Отключить цветной вывод. (Эта опция также передается portage.)

-d | --debug

Выводить большое количество дополнительной информации (на основе обработки -xv в bash)

-e | --exact

Устанавливать последние версии найденных пакетов, не принимая во внимание СЛОТ.

-h | --help

Вывести справку по синтаксису утилиты.

-i | --ignore

Удалить временные файлы, оставшиеся с предыдущих запусков программы.

-k | --keep-temp

Принудительно сохранять временные файлы даже после того, как revdep-rebuild успешно пересоберет пакеты. Несмотря на эту опцию, некорректные и устаревшие временные файлы БУДУТ удалены.

--library ИМЯ | -L ИМЯ

Искать обратные зависимости для определенной библиотеки или группы библиотек, а не для всех библиотек в системе. Будут установлены пакеты, использующие указанную библиотеку. ИМЯ может быть полным путем доступа к библиотеке либо регулярным выражением. (См. regex(7).)

-l | --no-ld-path

Не устанавливать LD_LIBRARY_PATH. **Примечание:** использование этой опции приведет к тому, что revdep-rebuild включит в журнал и некоторые файлы, в действительности не требующие переустановки.

-o | --no-order

Не проверять порядок сборки по списку глубоких зависимостей. В результате revdep-rebuild будет работать быстрее, но при сборке могут возникнуть ошибки. В этом случае попытайтесь запустить revdep-rebuild без опции -o и только потом оформляйте сообщение об ошибке на багтрекере.

-p | --pretend

"Холостой" запуск утилиты. Временные файлы не будут удалены. (Использование сразу двух опций, -k -p, избыточно, но не опасно.) При запуске revdep-rebuild не от root'a опция --pretend подразумевается.

-P | --no-progress

Отключить графическое отображение хода работы утилиты.

-q | --quiet

Уменьшить объем выводимой информации и отключить отображение хода работы. (Эта опция также передается portage.)

-v | --verbose

Увеличить объем выводимой информации. (Будет выведена среда поиска revdep-rebuild.)

Опции, стоящие после --, игнорируются revdep-rebuild и напрямую передаются emerge.

НАСТРОЙКА

В настоящее время revdep-rebuild более не использует жестко закодированные пути. Если вы хотите изменить стандартное поведение утилиты, редактируйте значения описанных ниже переменных.

LD_LIBRARY_MASK - Мaska особо анализируемых библиотек SEARCH_DIRS - Перечень каталогов, по которым будет осуществляться поиск исполняемых файлов и библиотек SEARCH_DIRS_MASK - Перечень каталогов, которые будут исключены из области поиска

Вы можете переопределить эти переменные, установив переменную в окружении перед установкой. Для этого следует включить соответствующую запись в /etc/make.conf или поместить файл с необходимыми переменными по адресу /etc/revdep-rebuild.

Переменныечитываются и устанавливаются в следующем порядке: настройки рабочей среды - временные изменения, внесенные пользователем

/etc/make.conf - постоянные изменения, внесенные пользователем

/etc/revdep-rebuild/* - постоянные изменения, внесенные авторами ебилда

Хотя пользователь может редактировать по своему усмотрению файлы в каталоге /etc/revdep-rebuild, имейте в виду, что данный каталог не защищен по умолчанию от изменений конфигурации - и, следовательно, лежащие в нем файлы могут быть удалены и/или перезаписаны при установке очередного ебилда. Чтобы изменить это поведение, добавьте /etc/revdep-rebuild в значение переменной CONFIG_PROTECT в файле /etc/make.conf. "-*" означает, что содержимое переменной будет очищено начиная с указанного места. Если, например, установлено SEARCH_DIRS="/usr/bin -*", то переменная SEARCH_DIRS будет содержать только /usr/bin

revdep-rebuild использует значения переменных NOCOLOR и PORTAGE_NICENESS, выставленные в файле /etc/make.conf

ПРИМЕРЫ

Перед полноценным запуском revdep-rebuild рекомендуется сначала выполнить следующее:
revdep-rebuild --ignore --pretend

Для поиска по всей системе, кроме каталогов /mnt и /home:

env SEARCH_DIRS="/ -*" SEARCH_DIRS_MASK="/mnt /home" revdep-rebuild

Для пересборки пакетов, которые зависят от libkdecor.so.4 из KDE 3.3:

revdep-rebuild --library /usr/kde/3.3/lib/libkdecor.so.4

Для пересборки пакетов, которые зависят от libImlib.so и libImlib2.so:

revdep-rebuild --library libImlib[2]*.so.*

ФАЙЛЫ

revdep-rebuild сохраняет ряд псевдо-временных файлов в каталоге /var/cache/revdep-rebuild/. Их удаление может повысить точность, но работа утилиты замедлится:

_0_env.rr_

Содержит переменные окружения

_1_files.rr_

Содержит список файлов, включенных в поиск

_2_ldpath.rr_

Содержит путь LDPATH

_3_broken.rr_

Содержит список "битых" файлов

_3_errors.rr_

Содержит вывод ошибок ldd

_4_raw.rr_

Содержит "сырой" список пакетов

_4_owners.rr_

Содержит список владельцев файлов

_4_pkgs.rr_

Содержит неотсортированные простые имена пакетов

_4_ebuilds.rr_

Содержит неотсортированные атомы

_5_order.rr_

Содержит отсортированные атомы

_6_status.rr_

Содержит вывод ошибок ldd

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

revdep-rebuild возвращает нуль при завершении, если и сама утилита, и **emerge** отработали успешно; в противном случае возвращается число, отличное от нуля.

ОШИБКИ

Об обнаруженных ошибках сообщайте на <http://bugs.gentoo.org> - но только в том случае, если речь не об ошибке сборки с опцией -o или -e. К сообщению приложите файлы из каталога /var/cache/revdep-rebuild/, вывод команды emerge --info... и ваши патчи. ;)

СМ. ТАКЖЕ

[emerge\(1\)](#), [portage\(5\)](#), [regex\(7\)](#)

ПЕРЕВОД

- Елена Гаврилова [\[e.vl.gavrilova@yandex.ru\]\(mailto:e.vl.gavrilova@yandex.ru\)\](mailto:e.vl.gavrilova@yandex.ru)

Справка по gentoolkit-dev

Gentoolkit-dev ([app-portage/gentoolkit-dev](#)) - набор скриптов в помощь разработчикам под Gentoo.

- [ebump](#) - Gentoo: сообщает номер ревизии ебилда
- [echangelog](#) - Gentoo: обновляет чейнджлоги (ChangeLogs) портежей
- [ekeyword](#) - Gentoo: изменяет ключевые слова (KEYWORDS) пакетов
- [eviewcvs](#) - Gentoo: генерирует URL для ViewCVS
- [imlate](#) - выводит предложения ключевых слов на основании заданной архитектуры

ebump

НАЗВАНИЕ

ebump - Gentoo: сообщает номер ревизии ебилда

СИНТАКСИС

ebump [*опция*] <_пакет\[-версия\]_>

ОПИСАНИЕ

Утилита *ebump* сообщает номер ревизии определенного ебилда и все вспомогательных файлов в каталоге *files/*, имеющих соответствующий суффикс версии.

По умолчанию все новые файлы ревизий добавляются в VCS.

Вы должны находиться в каталоге обрабатываемого ебилда.

ОПЦИИ

-C

--no-vcs

Не добавлять новые файлы в VCS.

-V

--version

Вывести информацию о версии и выйти.

-v

--verbose

Выводить более подробную информацию. Эта опция может использоваться неоднократно.

-q

--quiet

Выводить лишь самую необходимую информацию.

-a

--no-auxfiles

Не обрабатывать вспомогательные файлы (*files/**).

-c

--no-changelog

Не обновлять чейнджлоги (с помощью *echangelog*).

-m <_содержимое чейнджлога_>
--message <_содержимое_чейнджлога_>

Позволяет задать текст, который будет добавлен в чейнджлог вместо стандартного подстановочного.

-d

--delete-old

Удалить из VCS более раннюю ревизию и вспомогательные файлы. Это *небезопасно!* Не используйте данную опцию, если не вполне уверены в своих действиях, так как:

- 1) более ранняя ревизия, возможно, стабильно работает на архитектуре, отличной от вашей;
- 2) удаляемые вспомогательные файлы могут понадобиться другим версиям ебилда;
- 3) как правило, новая ревизия должна пройти тестирование, прежде чем она будет помечена как стабильная.

КОНФИГУРАЦИЯ

`/etc/gentoolkit/ebump.conf`
`~/.gentoo/ebump.conf`

Из этих файлов *ebump* черпает настройки.

opt_verbose (значение по умолчанию 1) - степень подробности выводимой информации, от 0 до 10

opt_add_changelog (значение по умолчанию *y*) - добавить запись в чейнджлог

opt_add_vcs (значение по умолчанию *y*) - добавить новые файлы в VCS

opt_bump_auxfiles (значение по умолчанию *y*) - обновить вспомогательные файлы в каталоге `files/`

opt_delete_old (значение по умолчанию *n*) - удалить прежнюю ревизию (**ОСТОРОЖНО!**)

opt_commitmessage (значение по умолчанию "") - сообщение чейнджлога по умолчанию

(УСТАРЕЛО)

`~/.gentoo/gentool-env` Из этого файла *ebump* берет переменные окружения **AUTHORNAME** и **AUTHOREMAIL**, которые используются для создания корректных записей ChangeLog.

СМ. ТАКЖЕ

Другие утилиты из комплекта *app-portage/gentoolkit-dev*, в частности, [echangeLog](#)(1) и [ekeyword](#)(1).

АВТОРЫ

- Karl Trygve Kalleberg [\[karltk@gentoo.org\]\(mailto:karltk@gentoo.org\)\](mailto:karltk@gentoo.org)
- Christian Ruppert [\[idl0r@gentoo.org\]\(mailto:idl0r@gentoo.org\)\](mailto:idl0r@gentoo.org)

ПЕРЕВОД

- Елена Гаврилова [\[e.vl.gavrilova@yandex.ru\]\(mailto:e.vl.gavrilova@yandex.ru\)\](mailto:e.vl.gavrilova@yandex.ru)

0.1.1

echangeLog

НАЗВАНИЕ

`echangeLog` - Gentoo: обновляет чейнджлоги (ChangeLogs) портежей

СИНТАКСИС

`echangeLog [текст]`

ОПИСАНИЕ

Эта утилита позволяет легко создавать и обновлять чейнджлоги (ChangeLogs) портежей в Gentoo. Она проверяет текущий каталог (предполагается, что это каталог пакета - например, /usr/portage/app-editors/vim), выявляет, какие файлы были изменены или добавлены, и по результатам проверки помещает запись в чейнджлог. Если *текст* в аргументе командной строки отсутствует, echangelog запросит его у пользователя.

Чтобы echangelog мог создать отчет об изменениях, они должны быть произведены до вызова утилиты. Например, вам следует предварительно выполнить cvs add применительно к вашим файлам, в противном случае echangelog не будет знать, что они являются частью обновления.

Если размер вашего текста превышает 80 символов в ширину, он будет автоматически переразбит на строки для удобства обращения к чейнджлогу. Если вам нужно специальное форматирование, то вы можете либо (1) запустить echangelog без текста в командной строке, убедившись, что ваша запись не слишком велика, либо (2) редактировать чейнджлог вручную. Если вы предпочитаете второй способ, мы рекомендуем вам использовать синтаксис вида "echangelog *что-нибудь*", чтобы избежать ошибок в заголовках, затем вручную произвести правку файла и заменить фрагмент *что-нибудь* на текст, который вам нужен в чейнджлоге.

Помимо обновления самого чейнджлога, echangelog автоматически обновляет дату выпуска всех устаревших ебилдов. Эти обновления включаются в перечень изменений, выводимый утилитой echangelog по завершении работы.

ОПЦИИ

В настоящее время утилита echangelog настолько проста, что не предусматривает опций. Возможно, в дальнейшем будут добавлены справочные --help и --version, но на данный момент достаточно для получения этой информации отслеживать версию gentoolkit.

ПРИМЕРЫ

Создать чейнджлог для совершенно нового пакета. Заголовок анализируется skel.ebuild.

```
$ cvs add metalog-0.1.ebuild
cvs server: use \*(Aqcvs commit\*(Aq to add this file permanently
```

```
$ echangelog \*(AqNew ebuild, thanks to Harvey McGillicuddy\*(Aq
--- ChangeLog 1969-12-31 19:00:00.000000000 -0500
+++ ChangeLog.new 2003-02-23 14:04:06.000000000 -0500
@@ -0,0 +1,9 @@
+# ChangeLog for app-admin/metalog
## Copyright 2000-2003 Gentoo Technologies, Inc.; Distributed under the
GPL v2
## $Header$
+
+*metalog-0.1 (23 Feb 2003)
+
+ 23 Feb 2003; Aron Griffis <agriffis@gentoo.org> metalog-0.1.ebuild :
+  New ebuild, thanks to Harvey McGillicuddy
+
```

Обновить до ревизии. Обратите внимание, что предварительно вам необходимо выполнить "cvs add" - в противном случае echangelog не увидит новый файл.

```
$ cvs add metalog-0.1-r1.ebuild
cvs server: use \*(Aqcvs commit\*(Aq to add this file permanently
```

```
$ echangeLog \*(AqBump revision to fix bug #999\*(Aq
--- ChangeLog 2003-02-23 14:04:06.000000000 -0500
+++ ChangeLog.new 2003-02-23 14:07:48.000000000 -0500
@@ -2,6 +2,11 @@
 # Copyright 2000-2003 Gentoo Technologies, Inc.; Distributed under the
GPL v2
 # $Header$


+*metalog-0.1-r1 (23 Feb 2003)
+
+ 23 Feb 2003; Aron Griffis <agriffis@gentoo.org> metalog-0.1-
r1.ebuild :
+  Bump revision to fix bug #999
+
*metalog-0.1 (23 Feb 2003)
```

```
23 Feb 2003; Aron Griffis <agriffis@gentoo.org> metalog-0.1.ebuild :
```

Если вам нужна запись в несколько строк, опустите текст в аргументе командной строки.

```
$ echangeLog
Please type the log entry, finish with ctrl-d
Bump revision to fix bug #999. Necessary to bump the revision because
the problem appears at run-time, not compile-time. This should also
give users the updated default configuration file.
--- ChangeLog 2003-02-23 14:09:12.000000000 -0500
+++ ChangeLog.new 2003-02-23 14:12:43.000000000 -0500
@@ -2,6 +2,13 @@
 # Copyright 2000-2003 Gentoo Technologies, Inc.; Distributed under the
GPL v2
 # $Header$


+*metalog-0.1-r1 (23 Feb 2003)
+
+ 23 Feb 2003; Aron Griffis <agriffis@gentoo.org> metalog-0.1-
r1.ebuild :
+  Bump revision to fix bug #999. Necessary to bump the revision because
+  the problem appears at run-time, not compile-time. This should also
+  give users the updated default configuration file.
+
*metalog-0.1 (23 Feb 2003)
```

```
23 Feb 2003; Aron Griffis <agriffis@gentoo.org> metalog-0.1.ebuild :
```

ПЕРЕМЕННЫЕ ОКРУЖЕНИЯ

ECHANGELOG_USER

Если echangelog не может определить ваше имя пользователя, чтобы сделать запись, вам нужно задать значение переменной ECHANGELOG_USER. Например, экспортировать ECHANGELOG_USER="Aron Griffis agriffis@gentoo.org"

ПРИМЕЧАНИЯ

В последних (на момент составления этого руководства) версиях echangelog помещает все новые записи в начало файла, а не ищет в нем соответствующую строку версии. Это связано с тем, что разработчики Gentoo придерживались разных мнений о новом формате чейнджлогов. Разумеется, использование двух разных форматов не может не вызвать проблемы.

Это означает, между прочим, что приведенные выше примеры несколько устарели, поскольку были скопированы из давней переписки. Тем не менее мы уверены, что они еще смогут быть вам полезны ;-)

АВТОРЫ

- Aron Griffis agriffis@gentoo.org.
Об обнаруженных ошибках сообщайте автору на <http://bugs.gentoo.org/>

ПЕРЕВОД

- Елена Гаврилова e.vl.gavrilova@yandex.ru

2009-04-28

EKEYWORD

НАЗВАНИЕ

ekeyword - Gentoo: изменяет ключевые слова (KEYWORDS) пакетов

СИНТАКСИС

ekeyword { архитектура|~архитектура|-архитектура|^архитектура } ебилды

ОПИСАНИЕ

Эта утилита позволяет легко добавлять или обновлять ключевые слова (KEYWORDS) в сете ебилдов. Аргументы в командной строке обрабатываются в порядке следования; таким образом, ключевые слова последовательно добавляются к текущему списку, и по мере этого обрабатываются ебилды.

Вы можете не указывать определенную архитектуру, а выставить "all", чтобы изменение коснулось всех текущих архитектур ебилда. Если вы поставите в начале символ ^, ekeyword удалит указанную архитектуру.

ОПЦИИ

В настоящее время утилита ekeyword настолько проста, что не предусматривает опций. Возможно, в дальнейшем будут добавлены справочные **--help** и **--version**, но на данный момент достаточно для получения этой информации отслеживать версию gentoolkit.

ПРИМЕРЫ

Пометить одну определенную архитектуру как стабильную:

```
$ ekeyword alpha metalog-0.7-r1.ebuild
metalog-0.7-r1.ebuild
-KEYWORDS="~alpha ~amd64 ~hppa ~ia64 ~mips ~ppc ~sparc ~x86"
```

```
+KEYWORDS="alpha ~amd64 ~hppa ~ia64 ~mips ~ppc ~sparc ~x86"
```

Обновляя пакет, пометить все пакеты как тестируемые:

```
$ ekeyword ~all metalog-0.7-r2.ebuild  
metalog-0.7-r2.ebuild  
-KEYWORDS="alpha amd64 hppa ia64 mips ppc sparc x86"  
+KEYWORDS="~alpha ~amd64 ~hppa ~ia64 ~mips ~ppc ~sparc ~x86"
```

Указать, что пакет выдает ошибку на всех архитектурах, кроме одной:

```
$ ekeyword ^all -* ~x86 metalog-0.7-r3.ebuild  
metalog-0.7-r3.ebuild  
-KEYWORDS="~alpha ~amd64 ~hppa ~ia64 ~mips ~ppc ~sparc ~x86"  
+KEYWORDS="-* ~x86"
```

Выполнить несколько операций сразу:

```
$ ekeyword alpha metalog-0.7-r1.ebuild  
~all metalog-0.7-r2.ebuild ^all -* ~x86 metalog-0.7-r3.ebuild  
metalog-0.7-r1.ebuild  
-KEYWORDS="~alpha ~amd64 ~hppa ~ia64 ~mips ~ppc ~sparc ~x86"  
+KEYWORDS="alpha ~amd64 ~hppa ~ia64 ~mips ~ppc ~sparc ~x86"  
metalog-0.7-r2.ebuild  
-KEYWORDS="alpha amd64 hppa ia64 mips ppc sparc x86"  
+KEYWORDS="~alpha ~amd64 ~hppa ~ia64 ~mips ~ppc ~sparc ~x86"  
metalog-0.7-r3.ebuild  
-KEYWORDS="~alpha ~amd64 ~hppa ~ia64 ~mips ~ppc ~sparc ~x86"  
+KEYWORDS="-* ~x86"
```

АВТОР

- Aron Griffis [\[agriffis@gentoo.org\]](mailto:agriffis@gentoo.org)(mailto:agriffis@gentoo.org)\.
Об обнаруженных ошибках сообщайте автору на <http://bugs.gentoo.org/>

ПЕРЕВОД

- Елена Гаврилова [\[e.vl.gavrilova@yandex.ru\]](mailto:e.vl.gavrilova@yandex.ru)(mailto:e.vl.gavrilova@yandex.ru)\

2009-08-30

EVIEWCVS

НАЗВАНИЕ

eviewcvs - Gentoo: генерирует URL для ViewCVS

СИНТАКСИС

eviewcvs [файлы]

ОПИСАНИЕ

Утилита генерирует список URL для ViewCVS на основе перечисленных в аргументе файлов или, если аргумента нет, файлов в текущем каталоге. Первая часть вывода (выделена зеленым цветом) представляет собой ссылки, позволяющие просмотреть сами файлы. Вторая часть вывода (выделена синим цветом) содержит ссылки, позволяющие отследить разницу между последней ревизией и текущим состоянием.

ОПЦИИ

В настоящее время утилита eviewcvs настолько проста, что не предусматривает опций. Возможно, в дальнейшем будут добавлены справочные **--help** и **--version**, но на данный момент достаточно для получения этой информации отслеживать версию gentoolkit.

ПРИМЕРЫ

Чтобы сгенерировать ссылки для ViewCVS для определенного файла, выполните:

```
$ eviewcvs package.mask
http://www.gentoo.org/cgi-bin/viewcvs.cgi/profiles/package.mask?
rev=1.3716&content-type=text/vnd.viewcvs-markup
http://www.gentoo.org/cgi-bin/viewcvs.cgi/profiles/package.mask?
r1=1.3715&r2=1.3716
```

Чтобы сгенерировать ссылки для ViewCVS для всех файлов в каталоге, выполните:

```
$ cd portage/net-misc/keychain
$ eviewcvs
http://sources.gentoo.org/viewcvs.py/gentoo-x86/net-
misc/keychain/ChangeLog?rev=1.54&view=markup
http://sources.gentoo.org/viewcvs.py/gentoo-x86/net-
misc/keychain/Manifest?rev=1.86&view=markup
http://sources.gentoo.org/viewcvs.py/gentoo-x86/net-
misc/keychain/keychain-2.6.1.ebuild?rev=1.3&view=markup
http://sources.gentoo.org/viewcvs.py/gentoo-x86/net-
misc/keychain/keychain-2.6.2.ebuild?rev=1.1&view=markup
http://sources.gentoo.org/viewcvs.py/gentoo-x86/net-
misc/keychain/metadata.xml?rev=1.3&view=markup
http://sources.gentoo.org/viewcvs.py/gentoo-x86/net-
misc/keychain/ChangeLog?r1=1.53&r2=1.54
http://sources.gentoo.org/viewcvs.py/gentoo-x86/net-
misc/keychain/Manifest?r1=1.85&r2=1.86
http://sources.gentoo.org/viewcvs.py/gentoo-x86/net-
misc/keychain/keychain-2.6.1.ebuild?r1=1.2&r2=1.3
http://sources.gentoo.org/viewcvs.py/gentoo-x86/net-
misc/keychain/metadata.xml?r1=1.2&r2=1.3
```

АВТОР

- Aron Griffis [agriffis@gentoo.org] (<mailto:agriffis@gentoo.org>)\nОб обнаруженных ошибках сообщайте автору на <http://bugs.gentoo.org/>

ПЕРЕВОД

- Елена Гаврилова [e.vl.gavrilova@yandex.ru] (<mailto:e.vl.gavrilova@yandex.ru>)\n

2009-05-06

imlate

НАЗВАНИЕ

imlate - выводит предложения ключевых слов на основании заданной архитектуры.

СИНТАКСИС

imlate [опции]

ОПЦИИ

--version

Вывести номер версии программы и выйти.

-h, --help

Показать эту справку и выйти.

-f ФАЙЛ, --file=ФАЙЛ

Записать результат в ФАЙЛ [по умолчанию будет использован стандартный вывод].

-m АРХИТЕКТУРА, --main=АРХИТЕКТУРА

Установить основную АРХИТЕКТУРу (например, архитектуру вашей системы) [по умолчанию - amd64].

-t АРХИТЕКТУРА, --target=АРХИТЕКТУРА

Выставить адресуемую архитектуру (например, x86) [значение по умолчанию - x86].

--mtime=ВРЕМЯ

Установить минимальное значение ВРЕМени изменений, в днях [значение по умолчанию - 30]

-s, --stable

Выводить предложения для стабильных пакетов (например, стандартным результатом будут -s и -k) [значение по умолчанию - True (истинно)].

-k, --keyword

Выводить предложения по ключевому слову (например, стандартным результатом будут -s и -k) [значение по умолчанию - True (истинно)].

-M МЕЙНТЕЙНЕР, --maintainer=МЕЙНТЕЙНЕР

Выводить только пакеты от указанного мейнтейнера.

-H ГРУППА, --herd=ГРУППА

Выводить только пакеты от указанной группы сопровождения.

-C КАТЕГОРИИ, --category=КАТЕГОРИИ, --categories=КАТЕГОРИИ

Включить в поиск указанную категорию/категории (через запятую) [по умолчанию - none (нет)].

АВТОРЫ

- Christian Ruppert [\(mailto:idl0r@gentoo.org\)](mailto:idl0r@gentoo.org)

ОШИБКИ

Об обнаруженных ошибках сообщайте на <http://bugs.gentoo.org>

ПЕРЕВОД

- Елена Гаврилова [\(mailto:e.vl.gavrilova@yandex.ru\)](mailto:e.vl.gavrilova@yandex.ru)

1.0.0