CPAN (Источник, CPAN) версия 2.36

Содержание

НАЗВАНИЕ

CPAN - запрашивать, загружать и создавать модули perl с сайтов CPAN

КРАТКИЙ ОБЗОР

Интерактивный режим:

```
perl -MCPAN -e shell
```

--или--

```
cpan
```

Основные команды:

```
# Modules:

cpan> install Acme::Meta                      # in the shell

CPAN::Shell->install("Acme::Meta");            # in perl

# Distributions:

cpan> install NWCLARK/Acme-Meta-0.02.tar.gz    # in the shell

CPAN::Shell->
  install("NWCLARK/Acme-Meta-0.02.tar.gz");    # in perl

# module objects:

$mo = CPAN::Shell->expandany($mod);
$mo = CPAN::Shell->expand("Module",$mod);      # same thing

# distribution objects:

$do = CPAN::Shell->expand("Module",$mod)->distribution;
$do = CPAN::Shell->expandany($distro);         # same thing
$do = CPAN::Shell->expand("Distribution",
                          $distro);            # same thing
```

ОПИСАНИЕ

Модуль CPAN автоматизирует или, по крайней мере, упрощает создание и установку модулей и расширений perl. Он включает в себя некоторые примитивные возможности поиска и знает, как использовать LWP, HTTP::Tiny, Net :: FTP и некоторые внешние клиенты загрузки для извлечения дистрибутивов из сети.

Они извлекаются с одного или нескольких зеркальных сайтов CPAN (всеобъемлющей сети архивов Perl) и распаковываются в выделенный каталог.

Модуль CPAN также поддерживает *пакеты* модулей с именами и версиями. Пакеты упрощают обработку наборов связанных модулей. Смотрите Пакеты ниже.

Пакет содержит диспетчер сеансов и диспетчер кэша. Диспетчер сеансов отслеживает, что было извлечено, создано и установлено в текущем сеансе. Менеджер кэша отслеживает дисковое пространство, занятое процессами make, и удаляет лишнее пространство с помощью простого механизма FIFO.

Все предоставляемые методы доступны в стиле программиста и в стиле интерактивной оболочки.

CPAN::shell([$prompt, $command]) Запуск интерактивного режима

Войдите в интерактивный режим, запустив

```
perl -MCPAN -e shell
```

или

```
cpan
```

который выводит вас в интерфейс readline. Если установлены `Term::ReadKey` и любой из `Term::ReadLine::Perl` или `Term::ReadLine::Gnu` , поддерживается журнал выполнения команд.

Один раз в командной строке введите `h` для получения одностраничной справки; остальное должно быть понятно само по себе.

Вызов функции `shell` принимает два необязательных аргумента: один - приглашение, второй - начальную командную строку по умолчанию (последнее работает, только если установлен реальный модуль интерфейса ReadLine).

Наиболее распространенными вариантами использования интерактивных режимов являются

Поиск авторов, пакетов, файлов дистрибутива и модулей

Существуют соответствующие однобуквенные команды `a`, `b`, `d`, и `m` для каждой из четырех категорий и еще одна, `i` для любой из упомянутых четырех. Каждая из четырех сущностей реализована как класс с немного отличающимися методами отображения объекта.

Аргументами этих команд являются либо строки, точно соответствующие строке идентификации объекта, либо регулярные выражения, сопоставленные без учета регистра с различными атрибутами объектов. Анализатор распознает регулярное выражение только тогда, когда вы заключаете его в косую черту.

Принцип заключается в том, что количество найденных объектов влияет на способ отображения элемента. Если поиск находит один объект, результат отображается с помощью довольно подробного метода `as_string`, но если найдено более одного объекта, каждый объект отображается с помощью краткого метода `as_glimpse`.

Примеры:

```
cpan> m Acme::MetaSyntactic
Module id = Acme::MetaSyntactic
    CPAN_USERID  BOOK (Philippe Bruhat (BooK) <[...]>)
    CPAN_VERSION 0.99
    CPAN_FILE    B/BO/BOOK/Acme-MetaSyntactic-0.99.tar.gz
    UPLOAD_DATE  2006-11-06
    MANPAGE      Acme::MetaSyntactic - Themed metasyntactic variables names
    INST_FILE    /usr/local/lib/perl/5.10.0/Acme/MetaSyntactic.pm
    INST_VERSION 0.99
cpan> a BOOK
Author id = BOOK
    EMAIL        [...]
    FULLNAME     Philippe Bruhat (BooK)
cpan> d BOOK/Acme-MetaSyntactic-0.99.tar.gz
Distribution id = B/BO/BOOK/Acme-MetaSyntactic-0.99.tar.gz
    CPAN_USERID  BOOK (Philippe Bruhat (BooK) <[...]>)
    CONTAINSMODS Acme::MetaSyntactic Acme::MetaSyntactic::Alias [...]
    UPLOAD_DATE  2006-11-06
cpan> m /lorem/
Module  = Acme::MetaSyntactic::loremipsum (BOOK/Acme-MetaSyntactic-0.99.tar.gz)
Module    Text::Lorem            (ADEOLA/Text-Lorem-0.3.tar.gz)
Module    Text::Lorem::More      (RKRIMEN/Text-Lorem-More-0.12.tar.gz)
Module    Text::Lorem::More::Source (RKRIMEN/Text-Lorem-More-0.12.tar.gz)
cpan> i /berlin/
Distribution    BEATNIK/Filter-NumberLines-0.02.tar.gz
Module  = DateTime::TimeZone::Europe::Berlin (DROLSKY/DateTime-TimeZone-0.7904.tar.gz)
Module    Filter::NumberLines    (BEATNIK/Filter-NumberLines-0.02.tar.gz)
Author          [...]
```

Примеры иллюстрируют несколько аспектов: первые три запроса нацелены непосредственно на модули, авторов или дистрибутивы и дают ровно один результат. Последние два используют регулярные выражения и дают несколько результатов. Последний предназначен для всех пакетов, модулей, авторов и дистрибутивов одновременно. Если доступно более одного результата, они печатаются в однострочном формате.

`get`, `make`, `test` `install`, `clean` ,, **модули или дистрибутивы**

Эти команды принимают любое количество аргументов и исследуют, что необходимо для выполнения действия. Обработка аргументов происходит следующим образом:

```
known module name in format Foo/Bar.pm   module
other embedded slash                     distribution
  - with trailing slash dot              directory
enclosing slashes                        regexp
known module name in format Foo::Bar     module
```

Если аргументом является имя файла дистрибутива (распознается по встроенным косым чертам), оно обрабатывается. Если это модуль, CPAN определяет файл распространения, в который включен этот модуль, и обрабатывает его, следуя любым зависимостям, указанным в META.yml или Makefile модуля.PL (это поведение контролируется параметром конфигурации `prerequisites_policy`). Если аргумент заключен в косые черты, он обрабатывается как регулярное выражение: он расширяется, и если результатом является отдельный объект (дистрибутив, комплект или модуль), этот объект обрабатывается.

Пример:

```
install Dummy::Perl                  # installs the module
install AUXXX/Dummy-Perl-3.14.tar.gz  # installs that distribution
install /Dummy-Perl-3.14/             # same if the regexp is unambiguous
```

get загружает файл дистрибутива и распаковывает его, make создает, test запускает набор тестов и install устанавливает.

Любой make или test выполняется без каких-либо условий. An

```
install <distribution_file>
```

также запускается без каких-либо условий. Но для

```
install <module>
```

CPAN проверяет, требуется ли установка, и печатает *актуальный модуль*, если файл дистрибутива, содержащий модуль, не нуждается в обновлении.

CPAN также отслеживает, что он сделал в текущем сеансе, и не пытается собрать пакет во второй раз, независимо от того, удалось это или нет. Тестовый запуск не повторяется, если тест был успешно выполнен ранее. То же самое для установочных запусков.

force Прагма может предшествовать другой команде (в настоящее время: get , make , test или install), чтобы выполнить команду с нуля и попытаться продолжить работу после определенных ошибок. Смотрите раздел ниже о force и fforce прагме.

notest Прагма пропускает тестовую часть в процессе сборки.

Пример:

```
cpan> notest install Tk
```

Команда clean приводит к

```
make clean
```

выполняется в рабочем каталоге дистрибутивного файла.

readme , perldoc , look **модуль или дистрибутив**

readme отображает файл README соответствующего дистрибутива. Look получает и разворачивает (если еще не сделано) файл дистрибутива, изменяет соответствующий каталог и открывает процесс дочерней оболочки в этом каталоге. perldoc отображает документацию pod модуля в формате html или обычного текста.

ls **автор**

ls **globbing_expression**

В первой форме перечислены все файлы рассылки в каталоге CPAN автора и ниже, которые хранятся в файлах КОНТРОЛЬНЫХ СУММ, распространяемых на CPAN. Список повторяется по подкаталогам.

Вторая форма ограничивает или расширяет выходные данные с помощью масштабирования оболочки, как в следующих примерах:

```
ls JV/make*
ls GSAR/*make*
ls */*make*
```

Последний пример работает очень медленно и выдает дополнительные индикаторы выполнения, которые нарушают согласованность результата.

Обратите внимание, что при глобировании отображаются только каталоги, которые запрашиваются явно, например FOO / * список не будет отображаться FOO/bar/Acme-Sthg-n.nn.tar.gz. Это может рассматриваться как ошибка, которая может быть исправлена в какой-нибудь будущей версии.

failed

Команда failed сообщает обо всех дистрибутивах, в одном из которых произошел сбой make , test или install по какой-либо причине в текущем сеансе командной оболочки.

Постоянство между сеансами

Если YAML или YAML::Syck модуль установлен, запись внутреннего состояния всех модулей записывается на диск после каждого шага. Файлы содержат сигнатуру текущей версии perl для последующего прочтения.

Если переменной configuration build_dir_reuse присвоено значение true, то CPAN.pm считывает собранные файлы YAML. Если сохраненная сигнатура соответствует запущенному в данный момент perl, сохраненное состояние загружается в память таким образом, что эффективно устанавливается постоянство между сеансами.

force и fforce прагма

Чтобы ускорить работу в сложных сценариях установки, CPAN.pm отслеживает, что уже сделано, и отказывается выполнять некоторые действия во второй раз. A get , a make и an install не повторяются. A test повторяется только в том случае, если предыдущий тест был неудачным. Диагностическое сообщение, когда CPAN.pm отказывается что-либо делать во второй раз, является одним из *уже было* unwrapped|made|tested successfully или чем-то подобным. Еще одна ситуация, когда CPAN отказывается действовать, - это install если соответствующая операция test не увенчалась успехом.

Во всех этих случаях пользователь может переопределить это упрямое поведение, добавив перед командой слово force, например:

```
cpan> force get Foo
cpan> force make AUTHOR/Bar-3.14.tar.gz
cpan> force test Baz
cpan> force install Acme::Meta
```

Каждая *принудительная* команда выполняется со стиранием соответствующей части ее памяти.

fforce Pragma - это вариант, эмулирующий force get , который стирает всю память с последующим указанным действием, фактически перезапуская всю процедуру получения / изготовления / тестирования / установки с нуля.

Lockfile

Интерактивные сеансы по умолчанию поддерживают файл блокировки ~/.cpan/.lock . Пакетные задания могут выполняться без файла блокировки и не мешать друг другу.

Оболочка предлагает работать в *пониженном режиме*, когда другой процесс удерживает файл блокировки. Это экспериментальная функция, которая еще не очень хорошо протестирована. В этом случае вторая оболочка не записывает файл истории, не использует файл метаданных и выдает другое приглашение.

Signals

CPAN.pm устанавливает обработчики сигналов для SIGINT и SIGTERM. Предполагается, что, находясь в cpan-shell, вы можете в любое время нажать ^C и вернуться к приглашению cpan-shell. SIGTERM приведет к очистке cpan-оболочки и выходу из цикла оболочки. Вы можете эмулировать эффект SIGTERM, отправив два последовательных SIGINTs, что обычно означает двойное нажатие ^C .

CPAN.pm игнорирует SIGPIPE. Если пользователь устанавливает inactivity_timeout , SIGALRM используется во время выполнения подпроцесса perl Makefile.PL или perl Build.PL . SIGALRM также используется при анализе версии модуля и управляется version_timeout .

CPAN::Оболочка

Команды, доступные в интерфейсе оболочки, являются методами в пакете CPAN::Shell. Если вы вводите команду командной строки, ваш ввод разделяется процедурой Text::ParseWords::shellwords(), которая действует так же, как и большинство командных команд. Первое слово интерпретируется как вызываемый метод, а остальные слова рассматриваются как аргументы метода. Строки продолжения поддерживаются путем завершения строки буквенной обратной косой чертой.

autobundle

autobundle записывает файл пакета в $CPAN::Config->{cpan_home}/Bundle каталог. Файл содержит список всех модулей, которые доступны из CPAN и в настоящее время установлены в @INC. Дубликаты каждого дистрибутива исключены. Имя файла пакета зависит от текущей даты и счетчика, например, *Bundle/Snapshot_2012_05_21_00.pm*. Это устанавливается повторно, если запустить cpan Bundle::Snapshot_2012_05_21_00 или установить Bundle::Snapshot_2012_05_21_00 из командной строки CPAN.

Возвращаемое значение: путь к записанному файлу.

hosts

Примечание: эта функция все еще находится в альфа-режиме и может измениться в будущих версиях CPAN.pm

Эта команда предоставляет статистический обзор последних действий по загрузке. Данные для этого собраны в файле YAML FTPstats.yml в вашем cpan_home каталоге. Если модуль YAML не настроен или YAML не установлен, статистика не предоставляется.

install_tested

Установите все дистрибутивы, которые были успешно протестированы, но еще не установлены. Смотрите также is_tested .

is_tested

Перечислите все каталоги сборки дистрибутивов, которые были успешно протестированы, но еще не установлены. Смотрите также install_tested .

mkmyconfig

mkmyconfig() записывает ваш собственный файл CPAN::MyConfig в ваш `~/.cpan/` каталог, чтобы вы могли сохранить свои собственные настройки вместо общесистемных.

r [Модуль|/Регулярное выражение/]...

сканирует текущую установку perl на наличие модулей, более новая версия которых доступна в CPAN, и предоставляет их список. При вызове без аргумента перечисляются все возможные обновления; при вызове с аргументами список фильтруется по модулям и регулярным выражениям, указанным в качестве аргументов.

Список выглядит примерно так:

```
Package namespace         installed    latest  in CPAN file
CPAN                        1.94_64    1.9600  ANDK/CPAN-1.9600.tar.gz
CPAN::Reporter               1.1801    1.1902  DAGOLDEN/CPAN-Reporter-1.1902.tar.gz
YAML                           0.70      0.73  INGY/YAML-0.73.tar.gz
YAML::Syck                     1.14      1.17  AVAR/YAML-Syck-1.17.tar.gz
YAML::Tiny                     1.44      1.50  ADAMK/YAML-Tiny-1.50.tar.gz
CGI                            3.43      3.55  MARKSTOS/CGI.pm-3.55.tar.gz
Module::Build::YAML            1.40      1.41  DAGOLDEN/Module-Build-0.3800.tar.gz
TAP::Parser::Result::YAML      3.22      3.23  ANDYA/Test-Harness-3.23.tar.gz
YAML::XS                       0.34      0.35  INGY/YAML-LibYAML-0.35.tar.gz
```

Он подавляет дубликаты в столбце `in CPAN file` таким образом, дистрибутивы со многими обновляемыми модулями перечислены только один раз.

Обратите внимание, что список не отсортирован.

недавняя *** ЭКСПЕРИМЕНТАЛЬНАЯ КОМАНДА***

Команда `recent` загружает список последних загрузок в CPAN и *медленно* отображает их. Во время выполнения команды $SIG {INT} завершает цикл после отображения текущего элемента.

Примечание: Для выполнения этой команды требуется установленный XML::LibXML.

Примечание: Вся эта команда в настоящее время является всего лишь взломом и, вероятно, изменится в будущих версиях CPAN.pm, но общий подход, скорее всего, останется.

Примечание: Смотрите также smoke

перекомпилировать

recompile() - это специальная команда, которая не принимает аргументов и выполняет цикл создания / тестирования / установки методом перебора для всех установленных динамически загружаемых расширений (также известных как модули XS) с использованием "force". Основная цель этой команды - завершить сетевую установку. Представьте, что у вас есть общее дерево исходных текстов для двух разных архитектур. Вы решаете выполнить полностью независимую новую установку. Вы начинаете с одной архитектуры с помощью файла пакета, созданного ранее. CPAN устанавливает весь пакет за вас, но когда вы пытаетесь повторить задание на второй архитектуре, CPAN отвечает `"Foo up to date"` сообщением для всех модулей. Итак, вы вызываете перекомпиляцию CPAN для второй архитектуры, и все готово.

Еще одно популярное применение `recompile` - это помощь на случай, если ваш perl нарушит двоичную совместимость. Если один из модулей, используемых CPAN, в свою очередь зависит от совместимости с двоичными файлами (поэтому вы не можете запускать команды CPAN), то вам следует попробовать модуль CPAN :: Nox для восстановления.

пакет отчетов | Дистрибутив | Модуль

`report` Команда временно включает `test_report` переменную config, затем выполняет `force test` команду с заданными аргументами.
`force` Pragma повторно запускает тесты и повторяет каждый шаг, который ранее мог завершиться неудачей.

smoke *** ЭКСПЕРИМЕНТАЛЬНАЯ КОМАНДА***

***** ПРЕДУПРЕЖДЕНИЕ: эта команда загружает и запускает программное обеспечение из CPAN на ваш компьютер с совершенно неизвестным статусом. Вам никогда не следует делать это со своей обычной учетной записью и лучше иметь для этого выделенный, хорошо отделенный и защищенный компьютер. *****

Команда `smoke` берет список последних загрузок в CPAN, предоставленный командой `recent` , и проверяет их все. Во время выполнения команды $SIG{INT} определяется как означающий, что текущий элемент должен быть пропущен.

Примечание: Вся эта команда в настоящее время является всего лишь взломом и, вероятно, изменится в будущих версиях CPAN.pm, но общий подход, скорее всего, останется.

Примечание: Смотрите также последние

обновить [Модуль |/Регулярное выражение/]...

`upgrade` Команда сначала выполняет `r` команду с заданными аргументами, а затем устанавливает новейшие версии всех модулей, которые были перечислены в ней.

Четыре `CPAN::*` класса: Author, Bundle, Module, Distribution

Хотя это может считаться внутренним, иерархия классов имеет значение как для пользователей, так и для программистов. CPAN.pm имеет дело с четырьмя классами, упомянутыми выше, и все эти классы используют набор методов. Действует классический единый полиморфизм. Объект метакласса регистрирует все объекты всех видов и индексирует их с помощью строки. Строки, ссылающиеся на объекты, имеют отдельное пространство имен (ну, не полностью разделенное):

```
      Namespace                         Class

words containing a "/" (slash)      Distribution
 words starting with Bundle::           Bundle
       everything else            Module or Author
```

Модули знают связанные с ними объекты распространения. Они всегда ссылаются на самый последний официальный релиз. Разработчики могут помечать свои выпуски как нестабильные версии для разработки (вставляя символ подчеркивания в номер версии модуля, который также будет отражен в названии дистрибутива при запуске 'make dist'), поэтому по умолчанию не всегда используется действительно самый популярный и новейший дистрибутив. Если модуль Foo распространяется на CPAN как в версии 1.23, так и в 1.23_90, CPAN.pm предлагает удобный способ установки версии 1.23, указав

```
install Foo
```

При этом будет установлен полный файл дистрибутива (скажем, BAR/Foo-1.23.tar.gz) со всеми сопутствующими материалами. Но если вы хотите установить версию 1.23_90, вам нужно знать, где находится файл дистрибутива в CPAN относительно каталога authors/ id /. Если автором является BAR, это может быть BAR/Foo-1.23_90.tar.gz; так что вам пришлось бы сказать

```
install BAR/Foo-1.23_90.tar.gz
```

Первый пример будет управляться объектом класса CPAN::Module, второй - объектом класса CPAN::Distribution.

Интеграция локальных каталогов

Примечание: эта функция все еще находится в альфа-режиме и может измениться в будущих версиях CPAN.pm

Объекты распространения обычно являются дистрибутивами из CPAN, но для объектов распространения проектов, хранящихся на локальном диске, также существует несколько вырожденный случай. Эти объекты распространения имеют то же имя, что и локальный каталог, и заканчиваются точкой. Точка сама по себе также допускается для текущего каталога на момент использования CPAN.pm. Все действия, такие как `make`, `test` и `install`, применяются непосредственно к этому каталогу. Это придает команде `cpan .` интересный штрих: в то время как обычная мантра установки модуля CPAN без него CPAN.pm является одним из

```
perl Makefile.PL                perl Build.PL
       ( go and get prerequisites )
make                            ./Build
make test                       ./Build test
make install                    ./Build install
```

команда `cpan .` выполняет все это сразу. Он определяет, какая из двух мантр подходит, извлекает и устанавливает все необходимые компоненты, рекурсивно обрабатывает их и, наконец, завершает установку модуля в текущем каталоге, будь то модуль CPAN или нет.

Типичный вариант использования - частные модули или рабочие копии проектов из удаленных репозиториев на локальном диске.

Перенаправление

Обычные символы перенаправления оболочки `|` и `>` распознаются оболочкой cpan **только в том случае, если они окружены пробелом.** Таким образом, передача данных на пейджер или перенаправление выходных данных в файл работает примерно так же, как в обычной оболочке, с условием, что вы должны вводить дополнительные пробелы.

Поддержка плагинов *** ЭКСПЕРИМЕНТАЛЬНЫЙ***

Плагины - это объекты, реализующие любой из восьми существующих методов:

```
pre_get
post_get
pre_make
post_make
pre_test
post_test
pre_install
post_install
```

Параметр конфигурации `plugin_list` содержит список строк вида

```
Modulename=arg0,arg1,arg2,arg3,...
```

например:

```
CPAN::Plugin::Flurb=dir,/opt/pkgs/flurb/raw,verbose,1
```

Во время выполнения каждый указанный плагин создается как одноэлементный объект путем выполнения эквивалента этого псевдокода:

```
my $plugin = <string representation from config>;
<generate Modulename and arguments from $plugin>;
my $p = $instance{$plugin} ||= Modulename->new($arg0,$arg1,...);
```

Сгенерированные синглтоны сохраняются с момента создания экземпляра до конца сеанса командной строки. <plugin_list> можно перенастроить в любое время во время выполнения. Пока оболочка cpan запущена, она проверяет все активированные плагины в каждой из 8 контрольных точек, перечисленных выше, и запускает соответствующий метод, если он реализован для этого объекта. Метод вызывается с активным объектом CPAN::Distribution, переданным в качестве аргумента.

НАСТРОЙКА

При первом использовании модуля CPAN в диалоговом окне конфигурации делается попытка определить пару параметров, специфичных для сайта. Результат диалога сохраняется в виде ссылки на хэш `$CPAN::Config` в файле CPAN/Config.pm.

Значения по умолчанию, определенные в файле CPAN/Config.pm, могут быть переопределены в файле, определяемом пользователем: CPAN/MyConfig.pm. Такой файл лучше всего размещать в `$HOME/.cpan/CPAN/MyConfig.pm` , потому что `$HOME/.cpan` добавляется в путь поиска модуля CPAN перед операторами use() или require() . Команда mkmyconfig записывает этот файл за вас.

Команда `o conf` имеет различные навороты:

поддержка завершения

Если у вас установлен модуль ReadLine, вы можете нажать TAB в любой точке командной строки и `o conf` вам предложат заполнить встроенные подкоманды и / или имена переменных конфигурации.

отображается справка: o справка по conf

Отображает краткую справку

отображение текущих значений: o conf [КЛЮЧ]

Отображает текущее значение (значения) для этой переменной конфигурации. Без КЛЮЧА отображает все подкоманды и переменные конфигурации.

Пример:

```
o conf shell
```

Если КЛЮЧ начинается и заканчивается косой чертой, строка между ними обрабатывается как регулярное выражение и отображаются только ключи, соответствующие этому регулярному выражению

Пример:

```
o conf /color/
```

изменение скалярных значений: o ЗНАЧЕНИЕ КЛЮЧА conf

Присваивает переменной config KEY значение VALUE . Пустую строку можно указать, как обычно в оболочках, с помощью `''` или `""`

Пример:

```
o conf wget /usr/bin/wget
```

изменение значений списка: o conf КЛАВИША SHIFT | UNSHIFT | PUSH | POP | SPLICE|LIST

Если имя переменной конфигурации заканчивается на `list`, это список. `o conf KEY shift` удаляет первый элемент списка, `o conf KEY pop` удаляет последний элемент списка. `o conf KEYS unshift LIST` добавляет список значений к списку, `o conf KEYS push LIST` добавляет список значений к списку.

Аналогично, `o conf KEY splice LIST` передает СПИСОК соответствующей команде splice.

Наконец, любой другой список аргументов принимается как новое значение списка для КЛЮЧЕВОЙ переменной, отбрасывающее предыдущее значение.

Примеры:

```
o conf urllist unshift http://cpan.dev.local/CPAN
o conf urllist splice 3 1
o conf urllist http://cpan1.local http://cpan2.local ftp://ftp.perl.org
```

возврат к сохраненным значениям по умолчанию: o conf

Возвращает все переменные конфигурации в состояние, указанное в сохраненном файле конфигурации.

сохранение конфигурации: o фиксация conf

Сохраняет все переменные конфигурации в текущий файл конфигурации (CPAN /Config.pm или CPAN / MyConfig.pm, который был загружен при запуске).

Диалоговое окно настройки можно запустить в любое время позже снова, выполнив команду `o conf init` в командной строке CPAN.
Подмножество диалогового окна конфигурации можно запустить, выдав команду `o conf init WORD` где WORD - это любая допустимая переменная конфигурации или регулярное выражение.

Конфигурационные переменные

В настоящее время определены следующие ключи в ссылке на хэш $CPAN::Config:

```
allow_installing_module_downgrades
                    allow or disallow installing module downgrades
allow_installing_outdated_dists
                    allow or disallow installing modules that are
                    indexed in the cpan index pointing to a distro
                    with a higher distro-version number
applypatch          path to external prg
auto_commit         commit all changes to config variables to disk
build_cache         size of cache for directories to build modules
build_dir           locally accessible directory to build modules
build_dir_reuse     boolean if distros in build_dir are persistent
build_requires_install_policy
                    to install or not to install when a module is
                    only needed for building. yes|no|ask/yes|ask/no
bzip2               path to external prg
cache_metadata      use serializer to cache metadata
check_sigs          if signatures should be verified
cleanup_after_install
                    remove build directory immediately after a
                    successful install and remember that for the
                    duration of the session
colorize_debug      Term::ANSIColor attributes for debugging output
colorize_output     boolean if Term::ANSIColor should colorize output
colorize_print      Term::ANSIColor attributes for normal output
colorize_warn       Term::ANSIColor attributes for warnings
commandnumber_in_prompt
                    boolean if you want to see current command number
commands_quote      preferred character to use for quoting external
                    commands when running them. Defaults to double
                    quote on Windows, single tick everywhere else;
                    can be set to space to disable quoting
connect_to_internet_ok
                    whether to ask if opening a connection is ok before
                    urllist is specified
cpan_home           local directory reserved for this package
curl                path to external prg
dontload_hash       DEPRECATED
dontload_list       arrayref: modules in the list will not be
                    loaded by the CPAN::has_inst() routine
ftp                 path to external prg
ftp_passive         if set, the environment variable FTP_PASSIVE is set
                    for downloads
ftp_proxy           proxy host for ftp requests
ftpstats_period     max number of days to keep download statistics
ftpstats_size       max number of items to keep in the download statistics
getcwd              see below
gpg                 path to external prg
gzip                location of external program gzip
halt_on_failure     stop processing after the first failure of queued
                    items or dependencies
histfile            file to maintain history between sessions
histsize            maximum number of lines to keep in histfile
http_proxy          proxy host for http requests
inactivity_timeout breaks interactive Makefile.PLs or Build.PLs
                    after this many seconds inactivity. Set to 0 to
                    disable timeouts.
index_expire        refetch index files after this many days
inhibit_startup_message
                    if true, suppress the startup message
keep_source_where   directory in which to keep the source (if we do)
load_module_verbosity
                    report loading of optional modules used by CPAN.pm
lynx                path to external prg
make                location of external make program
make_arg            arguments that should always be passed to 'make'
make_install_make_command
                    the make command for running 'make install', for
                    example 'sudo make'
make_install_arg    same as make_arg for 'make install'
makepl_arg          arguments passed to 'perl Makefile.PL'
mbuild_arg          arguments passed to './Build'
mbuild_install_arg arguments passed to './Build install'
mbuild_install_build_command
                    command to use instead of './Build' when we are
                    in the install stage, for example 'sudo ./Build'
mbuildpl_arg        arguments passed to 'perl Build.PL'
ncftp               path to external prg
```

```
ncftpget           path to external prg
no_proxy           don't proxy to these hosts/domains (comma separated list)
pager              location of external program more (or any pager)
password           your password if you CPAN server wants one
patch              path to external prg
patches_dir        local directory containing patch files
perl5lib_verbosity verbosity level for PERL5LIB additions
plugin_list        list of active hooks (see Plugin support above
                   and the CPAN::Plugin module)
prefer_external_tar
                   per default all untar operations are done with
                   Archive::Tar; by setting this variable to true
                   the external tar command is used if available
prefer_installer   legal values are MB and EUMM: if a module comes
                   with both a Makefile.PL and a Build.PL, use the
                   former (EUMM) or the latter (MB); if the module
                   comes with only one of the two, that one will be
                   used no matter the setting
prerequisites_policy
                   what to do if you are missing module prerequisites
                   ('follow' automatically, 'ask' me, or 'ignore')
                   For 'follow', also sets PERL_AUTOINSTALL and
                   PERL_EXTUTILS_AUTOINSTALL for "--defaultdeps" if
                   not already set
prefs_dir          local directory to store per-distro build options
proxy_user         username for accessing an authenticating proxy
proxy_pass         password for accessing an authenticating proxy
pushy_https        use https to cpan.org when possible, otherwise use http
                   to cpan.org and issue a warning
randomize_urllist  add some randomness to the sequence of the urllist
recommends_policy  whether recommended prerequisites should be included
scan_cache         controls scanning of cache ('atstart', 'atexit' or 'never')
shell              your favorite shell
show_unparsable_versions
                   boolean if r command tells which modules are versionless
show_upload_date   boolean if commands should try to determine upload date
show_zero_versions boolean if r command tells for which modules $version==0
suggests_policy    whether suggested prerequisites should be included
tar                location of external program tar
tar_verbosity      verbosity level for the tar command
term_is_latin      deprecated: if true Unicode is translated to ISO-8859-1
                   (and nonsense for characters outside latin range)
term_ornaments     boolean to turn ReadLine ornamenting on/off
test_report        email test reports (if CPAN::Reporter is installed)
trust_test_report_history
                   skip testing when previously tested ok (according to
                   CPAN::Reporter history)
unzip              location of external program unzip
urllist            arrayref to nearby CPAN sites (or equivalent locations)
urllist_ping_external
                   use external ping command when autoselecting mirrors
urllist_ping_verbose
                   increase verbosity when autoselecting mirrors
use_prompt_default set PERL_MM_USE_DEFAULT for configure/make/test/install
use_sqlite         use CPAN::SQLite for metadata storage (fast and lean)
username           your username if you CPAN server wants one
version_timeout    stops version parsing after this many seconds.
                   Default is 15 secs. Set to 0 to disable.
wait_list          arrayref to a wait server to try (See CPAN::WAIT)
wget               path to external prg
yaml_load_code     enable YAML code deserialisation via CPAN::DeferredCode
yaml_module        which module to use to read/write YAML files
```

Вы можете установить и запросить каждый из этих параметров в интерактивном режиме в командной строке cpan с помощью `o conf` или `o conf init` команды, как указано ниже.

`o conf <scalar option>`

выводит текущее значение *скалярного параметра*

`o conf <scalar option> <value>`

Устанавливает значение для *скалярной опции* равным *value*

`o conf <list option>`

выводит текущее значение *параметра списка* в формате neatvalue от MakeMaker.

```
o conf <list option> [shift|pop]
```

сдвигает или выталкивает массив в переменной *опции списка*

```
o conf <list option> [unshift|push|splice] <list>
```

работает аналогично соответствующим командам perl.

интерактивное редактирование: o conf init [СООТВЕТСТВИЕ | СПИСОК]

Запускает интерактивное диалоговое окно конфигурации для сопоставления переменных. Без аргумента запускает диалоговое окно для всех поддерживаемых переменных конфигурации. Чтобы указать СООТВЕТСТВИЕ, аргумент должен быть заключен в косую черту.

Примеры:

```
o conf init ftp_passive ftp_proxy
o conf init /color/
```

Примечание: этот метод установки конфигурационных переменных часто дает больше объяснений о функционировании переменной, чем manpage.

CPAN::anycwd($path): Примечание к конфигурационной переменной getcwd

CPAN.pm часто изменяет текущий рабочий каталог и нуждается в определении собственного текущего рабочего каталога. По умолчанию он использует Cwd ::cwd, но если по какой-либо причине это не работает в вашей системе, настройте альтернативные варианты в соответствии со следующей таблицей:

cwd

Вызывает Cwd::cwd

getcwd

Вызывает Cwd::getcwd

fastcwd

Вызывает Cwd::fastcwd

getdcwd

Вызывает Cwd::getdcwd

backtickcwd

Вызывает внешнюю команду cwd.

Обратите внимание на формат параметра urllist

параметры списка URL-адресов соответствуют URL-адресам в соответствии с RFC 1738. Мы немного угадываем, не соответствует ли ваш URL-адрес, но если у вас возникли проблемы с `file` URL-адресами, пожалуйста, попробуйте правильный формат. Либо:

```
file://localhost/whatever/ftp/pub/CPAN/
```

или

```
file:///home/ftp/pub/CPAN/
```

Параметр urllist поддерживает CD-ROM

`urllist` Параметр таблицы конфигурации содержит список URL-адресов, используемых для загрузки. Если список содержит какие-либо `file` URL-адреса, CPAN всегда пытается сначала найти их. Эта функция отключена для индексных файлов. Итак, рекомендация для владельца компакт-диска с содержимым CPAN такова: включите свой локальный, возможно устаревший компакт-диск в качестве `file` URL-адреса в конце списка url-адресов, например

```
o conf urllist push file://localhost/CDROM/CPAN
```

CPAN.pm затем извлекает индексные файлы с одного из сайтов CPAN, которые находятся в начале списка url-адресов. Позже он проверит для каждого модуля, существует ли локальная копия самой последней версии.

Еще одна особенность urllist заключается в том, что сайт, с которого мы смогли успешно получить последний файл, автоматически получает токен предпочтения и используется в качестве первого сайта для следующего запроса. Таким образом, если вы добавите новый сайт во время выполнения, может случиться так, что ранее предпочитаемый сайт будет опробован в другой раз. Это означает, что если вы хотите запретить сайт для следующего переноса, он должен быть явно удален из списка url-адресов.

Сохранение параметра urllist

Если у вас установлен YAML.pm (или какой-либо другой модуль YAML, настроенный в `yaml_module`), CPAN.pm собирает несколько статистических данных о последних загрузках. Вы можете просмотреть статистику с помощью `hosts` команды или просмотреть их напрямую, заглянув в `FTPstats.yml` файл в вашем `cpan_home` каталоге.

Для получения некоторой интересной статистики рекомендуется установить параметр `randomize_urllist`; это вносит некоторую случайность в выбор URL.

Объявления зависимостей `requires` и `build_requires`

Начиная с CPAN.pm версии 1.88_51 модули, объявленные как `build_requires` в дистрибутиве, обрабатываются по-разному в зависимости от переменной конфигурации `build_requires_install_policy`. При значении `build_requires_install_policy` в `no` такой модуль не устанавливается. Он только собирается и тестируется, а затем сохраняется в списке протестированных, но удаленных модулей. Как таковой, он доступен во время сборки зависимого модуля путем интеграции пути к каталогам `blib/arch` и `blib/lib` в переменной среды PERL5LIB. Если `build_requires_install_policy` установлено значение `yes`, то оба модуля, объявленные как `requires` и те, которые объявлены как `build_requires`, обрабатываются одинаково. Установив значение `ask/yes` или `ask/no`, CPAN.pm запрашивает пользователя и соответствующим образом устанавливает значение по умолчанию.

Настройка параметров allow_installing_*

`allow_installing_*` Параметры оцениваются на этапе `make`. Если установлено значение `yes`, они разрешают тестирование и установку текущего дистрибутива и в остальном не оказывают никакого эффекта. Если установлено значение `no`, они могут прервать сборку (препятствуя тестированию и установке), в зависимости от содержимого `blib/` каталога. `blib/` Каталог - это каталог, в котором хранятся все файлы, которые обычно устанавливаются на этапе `install`.

`allow_installing_outdated_dists` сравнивает `blib/` каталог с индексом CPAN. Если он находит там что-то, что, согласно индексу, принадлежит другому dist, он прерывает текущую сборку.

`allow_installing_module_downgrades` сравнивает `blib/` каталог с уже установленными модулями, фактически с номерами их версий, определенными ExtUtils::MakeMaker или эквивалентными. Если устанавливаемый модуль понизит рейтинг уже установленного модуля, текущая сборка прерывается.

Интересный поворот происходит, когда документ distroprefs требует установки устаревшего dist через goto, в то время как `allow_installing_outdated_dists` запрещает это. Без дополнительных условий это привело бы к тому, что `allow_installing_outdated_dists` выиграли, а дистрибутивы проиграли. Таким образом, правильным решением в таком случае является написание второго документа distroprefs для дистрибутива, который `goto` указывает на `cpanconfig` там и отменяет его. Например.:

```
---
match:
  distribution: "^MAUKE/Keyword-Simple-0.04.tar.gz"
goto: "MAUKE/Keyword-Simple-0.03.tar.gz"
---
match:
  distribution: "^MAUKE/Keyword-Simple-0.03.tar.gz"
cpanconfig:
  allow_installing_outdated_dists: yes
```

Настройка для отдельных дистрибутивов (*Distroprefs*)

(**Примечание:** Эта функция была введена в версии CPAN.pm 1.8854)

Дистрибутивы на CPAN обычно ведут себя в соответствии с тем, что мы называем мантрой CPAN. Или с момента появления Module::Build мы должны говорить о двух мантрах:

```
perl Makefile.PL     perl Build.PL
make                 ./Build
make test            ./Build test
make install         ./Build install
```

Но некоторые модули не могут быть созданы с помощью этой мантры. Они пытаются получить некоторые дополнительные данные от пользователя через среду, дополнительные аргументы или в интерактивном режиме - таким образом нарушая установку больших пакетов, таких как Phalanx100, или модулей со многими зависимостями, таких как Plagger.

Система distroprefs `CPAN.pm` решает эту проблему, позволяя пользователю указывать дополнительную информацию и рецепты в файлах YAML, чтобы либо

- передайте дополнительные аргументы одной из четырех команд,
- настройка переменных среды
- создание экземпляра объекта Expect, который считывает данные с консоли, ожидает некоторых регулярных выражений и вводит некоторые ответы
- временное переопределение различных `CPAN.pm` переменных конфигурации

- укажите зависимости, которые забыл исходный разработчик
- полностью отключите установку объекта

Смотрите примеры файлов YAML и Data::Dumper, которые поставляются с `CPAN.pm` дистрибутивом в `distroprefs/` каталоге.

Имена файлов

Сами файлы YAML должны иметь `.yml` расширение; все остальные файлы игнорируются (для двух исключений см. *Резервные данные::Dumper и Storable* ниже). Содержащий каталог может быть указан в `CPAN.pm` в `prefs_dir` переменной config. Попробуйте `o conf init prefs_dir` в CPAN shell установить и активировать систему distroprefs.

Каждый файл YAML может содержать произвольные документы в соответствии со спецификацией YAML, и каждый документ обрабатывается как объект, который может указывать обработку отдельного дистрибутива.

Имена файлов можно выбирать произвольно; `CPAN.pm` всегда считывает все файлы (в алфавитном порядке) и принимает ключ `match` (см. Ниже в *языковых спецификациях*) в качестве хэш-ссылки, содержащей критерии соответствия, которые определяют, соответствует ли текущий дистрибутив документу YAML или нет.

Резервные данные:: Выгружаемые и хранимые

Если ни настроенный `yaml_module` , ни YAML.pm не установлен, CPAN.pm вернитесь к использованию Data::Dumper и Storable и ищите файлы с расширениями `.dd` or `.st` в `prefs_dir` каталоге. Ожидается, что эти файлы будут содержать одну или несколько хэш-ссылок. Ожидается, что для файлов, сгенерированных Data::Dumper, это будет сделано с помощью определения `$VAR1` , `$VAR2` и т.д. Оболочка YAML создаст их с помощью команды

```
ysh < somefile.yml > somefile.dd
```

Для сохраняемых файлов правило таково, что они должны быть сконструированы таким образом, чтобы `Storable::retrieve(file)` возвращала ссылку на массив, а элементы массива представляли по одному объекту distropref каждый. Преобразование из YAML будет выглядеть следующим образом:

```
perl -MYAML=LoadFile -MStorable=nstore -e '
    @y=LoadFile(shift);
    nstore(\@y, shift)' somefile.yml somefile.st
```

В ситуациях начальной загрузки обычно достаточно перевести всего несколько файлов YAML в Data::Dumper для таких важных модулей, как `YAML::Syck` , `YAML.pm` и `Expect.pm` . Если вы предпочитаете Storable, а не Data ::Dumper, не забудьте выбрать Storable версию, которая записывает более старый формат, чем все другие Storable версии, которые понадобятся для их чтения.

Схема действий

The following example contains all supported keywords and structures with the exception of `eexpect` which can be used instead of `expect` .

```
---
comment: "Demo"
match:
  module: "Dancing::Queen"
  distribution: "^CHACHACHA/Dancing-"
  not_distribution: "\.zip$"
  perl: "/usr/local/cariba-perl/bin/perl"
  perlconfig:
    archname: "freebsd"
    not_cc: "gcc"
  env:
    DANCING_FLOOR: "Shubiduh"
disabled: 1
cpanconfig:
  make: gmake
pl:
  args:
    - "--somearg=specialcase"

  env: {}

  expect:
    - "Which is your favorite fruit"
    - "apple\n"

make:
  args:
    - all
    - extra-all

  env: {}

  expect: []

  commandline: "echo SKIPPING make"

test:
  args: []

  env: {}

  expect: []

install:
  args: []

  env:
    WANT_TO_INSTALL: YES

  expect:
    - "Do you really want to install"
    - "y\n"

patches:
  - "ABCDE/Fedcba-3.14-ABCDE-01.patch"

depends:
  configure_requires:
    LWP: 5.8
  build_requires:
    Test::Exception: 0.25
  requires:
    Spiffy: 0.30
```

Языковые спецификации

Каждый документ YAML представляет собой единственную ссылку на хэш. Допустимые ключи в этом хэше следующие:

комментарий [скалярный]

Комментарий

cpanconfig [хэш]

Временно переопределять различные CPAN.pm переменные конфигурации.

Поддерживаются: build_requires_install_policy , check_sigs , make , make_install_make_command , prefer_installer , test_report . Пожалуйста, сообщайте об ошибке, когда вам нужен еще один поддерживаются.

зависит [хэш] * ЭКСПЕРИМЕНТАЛЬНАЯ ФУНКЦИЯ *****

Все три типа, а именно configure_requires , build_requires и requires поддерживаются способом, указанным в спецификации META.yml. Текущая реализация *объединяет* указанные зависимости с зависимостями, объявленными сопровождающим пакета. В будущей реализации это может быть изменено, чтобы переопределить исходное объявление.

отключено [логическое значение]

Указывает, что это распространение вообще не должно обрабатываться.

функции [массив] * ЭКСПЕРИМЕНТАЛЬНАЯ ФУНКЦИЯ *****

Экспериментальная реализация для работы с optional_features из META.yml. Все еще требует согласования с программным обеспечением установщика и в настоящее время работает только для объявления META.yml dynamic_config=0 . Используйте с осторожностью.

переход [строка]

Каноническое название делегированного дистрибутива для установки вместо него. Полезно, когда новая версия, хотя сама тестируется нормально, что-то нарушает, или уже загружен релиз разработчика или форк, который лучше последней выпущенной версии.

установить [хэш]

Инструкции по обработке для фазы CPAN mantra make install или ./Build install . Смотрите ниже в разделе *Инструкции по обработке*.

создать [хэш]

Инструкции по обработке для фазы CPAN mantra make или ./Build . Смотрите ниже в разделе *Инструкции по обработке*.

совпадение [хэш]

Хэш-ссылка с одним или несколькими ключами distribution , module , perl , perlconfig , env , и,,, которые указывают, предназначен ли документ для определенного дистрибутива или установки CPAN. Ключи с префиксом not_ отменяют соответствующее совпадение.

Соответствующие значения интерпретируются как регулярные выражения. distribution Связанное значение будет сопоставлено с каноническим именем дистрибутива, например "AUTHOR/Foo-Bar-3.14.tar.gz".

module Связанный модуль будет сопоставляться со *всеми* модулями, содержащимися в дистрибутиве, пока не будет совпадать один модуль.

perl Связанный будет сопоставлен с $^X (но с указанием абсолютного пути).

Значение, связанное с perlconfig , само по себе является hashref, которое сопоставляется с соответствующими значениями в %Config::Config хэше, находящемся в Config.pm модуле. Ключи с префиксом not_ отменяют соответствующее совпадение.

Значение, связанное с env , само по себе является hashref, которое сопоставляется с соответствующими значениями в %ENV хэше. Ключи с префиксом not_ отрицает соответствующее совпадение.

Если указано более одного ограничения module , distribution и т.д., все результаты отдельно вычисленных значений соответствия должны совпадать. Если это так, то hashref, представленный документом YAML, возвращается в качестве структуры предпочтений для текущего дистрибутива.

исправления [массив]

Массив исправлений в CPAN или на локальном диске, которые необходимо применять по порядку с помощью внешней программы исправления. Если значение для -p параметра равно 0 или 1 определяется путем предварительного прочтения исправления. Путь к каждому исправлению является либо абсолютным путем в локальной файловой системе, либо относительным к каталогу исправлений, указанному в patches_dir переменной конфигурации, либо в формате канонического имени дистрибутива. Для получения примеров, пожалуйста, обратитесь к каталогу distroprefs / в дистрибутиве CPAN.pm (эти примеры не установлены по умолчанию).

Примечание: если applypatch программа установлена и CPAN::Config знает об этом , и makepatch программа написала исправление, то CPAN.pm давайте applypatch применим исправление. Оба makepatch и applypatch доступны на CPAN в JV/makepatch-* дистрибутиве.

pl [хэш]

Инструкции по обработке для фазы CPAN mantra perl Makefile.PL или perl Build.PL . Смотрите ниже в разделе *Инструкции по обработке*.

тест [хэш]

Инструкции по обработке для фазы CPAN mantra make test или ./Build test . Смотрите ниже в разделе *Инструкции по обработке*.

Инструкции по обработке

аргументы [массив]

Аргументы, которые необходимо добавить в командную строку

командная строка

Полная командная строка для запуска через `system()`. Во время выполнения переменной среды PERL присваивается значение $ ^ X (но с указанием абсолютного пути). Если `commandline` указано, `args` не используется.

eexpect [хэш]

Расширено `expect`. Это ссылка на хэш с четырьмя разрешенными ключами, `mode`, `timeout`, `reuse` и `talk`.

Вы должны установить `Expect` модуль для использования `eexpect`. CPAN.pm не устанавливает его за вас.

`mode` может иметь значения `deterministic` для случая, когда все вопросы задаются в записанном порядке, и `anyorder` для случая, когда вопросы могут задаваться в любом порядке. Режим по умолчанию - `deterministic`.

`timeout` время ожидания указывается в секундах. Время ожидания с плавающей запятой в порядке. С помощью `mode=deterministic` тайм-аут обозначает время ожидания для каждого вопроса; с помощью `mode=anyorder` он обозначает время ожидания для каждого байта, полученного из потока или вопросов.

`talk` это ссылка на массив, содержащий чередующиеся вопросы и ответы. Вопросы представляют собой регулярные выражения, а ответы - буквенные строки. Модуль Expect отслеживает поток выполнения внешней программы (`perl Makefile.PL`, `perl Build.PL`, `make` и т.д.).

Для `mode=deterministic`, the CPAN.pm вводит соответствующий ответ, как только поток соответствует регулярному выражению.

Для `mode=anyorder` CPAN.pm отвечает на вопрос, как только истекает время ожидания для следующего байта во входном потоке. В этом режиме вы можете использовать параметр `reuse`, чтобы решить, что произойдет с парой вопрос-ответ после ее использования. В случае по умолчанию (reuse= 0) они удаляются из массива, чтобы избежать случайного повторного использования. Если вы хотите ответить на вопрос `Do you really want to do that` несколько раз, то он должен включаться в массив по крайней мере так часто, как вы хотите, чтобы этот ответ давался. Установка параметра `reuse` равным 1 делает это повторение ненужным.

env [хэш]

Переменные среды, которые необходимо задать во время выполнения команды

ожидать [массив]

Вы должны установить `Expect` модуль для использования `expect`. CPAN.pm не устанавливает его за вас.

`expect: <array>` краткое обозначение для этого `eexpect`:

```
eexpect:
        mode: deterministic
        timeout: 15
        talk: <array>
```

Schema verification with `Kwalify`

Если у вас установлен `Kwalify` модуль (который является частью пакета ::CPANxxl), то все ваши файлы distroprefs проверяются на синтаксическую корректность.

Примеры файлов Distroprefs

`CPAN.pm` поставляется с коллекцией примеров файлов YAML. Обратите внимание, что на самом деле это всего лишь примеры, и их не следует использовать без осторожности, поскольку они не могут соответствовать целям каждого пользователя. В конце концов, авторам пакетов, которые задают вопросы, нужно было задавать вопросы, поэтому вам следует следить за их вопросами и адаптировать примеры к вашей среде и вашим потребностям. Вы были предупреждены :-)

ИНТЕРФЕЙС ПРОГРАММИСТА

Если вы не входите в оболочку, команды оболочки доступны как методы (`CPAN::Shell->install(...)`), так и функции в вызывающем пакете (`install(...)`). Перед вызовом низкоуровневых команд имеет смысл инициализировать нужные вам компоненты CPAN, например:

```
CPAN::HandleConfig->load;
CPAN::Shell::setup_output;
CPAN::Index->reload;
```

Высокоуровневые команды выполняют такие инициализации автоматически.

В настоящее время существует только один класс со стабильным интерфейсом - CPAN::Shell. Все команды, доступные в CPAN shell, являются методами класса CPAN::Shell. Аргументы в командной строке передаются в качестве аргументов метода.

Итак, если вы возьмете, к примеру, команду оболочки

```
notest install A B C
```

фактически выполняемая команда

```
CPAN::Shell->notest("install","A","B","C");
```

Каждая из команд, создающих списки модулей (`r` , `autobundle` , `u`), также возвращает список идентификаторов всех модулей в списке.

развернуть ($type,@things)

Идентификаторы всех объектов, доступных в программе, представляют собой строки, которые могут быть расширены до соответствующих реальных объектов с помощью `CPAN::Shell->expand("Module",@things)` метода. Expand возвращает список объектов CPAN::Module в соответствии с `@things` приведенными аргументами. В скалярном контексте он возвращает только первый элемент списка.

расширение (@things)

Похоже на expand, но возвращает объекты соответствующего типа, т.е. Объекты CPAN::Bundle для пакетов, объекты CPAN::Module для модулей и объекты CPAN::Distribution для дистрибутивов. Примечание: он не распространяется на объекты CPAN::Author.

Примеры программирования

Это позволяет программисту выполнять операции, сочетающие функциональные возможности, доступные в командной оболочке.

```
# install everything that is outdated on my disk:
perl -MCPAN -e 'CPAN::Shell->install(CPAN::Shell->r)'

# install my favorite programs if necessary:
for $mod (qw(Net::FTP Digest::SHA Data::Dumper)) {
    CPAN::Shell->install($mod);
}

# list all modules on my disk that have no VERSION number
for $mod (CPAN::Shell->expand("Module","/./")) {
    next unless $mod->inst_file;
    # MakeMaker convention for undefined $VERSION:
    next unless $mod->inst_version eq "undef";
    print "No VERSION in ", $mod->id, "\n";
}

# find out which distribution on CPAN contains a module:
print CPAN::Shell->expand("Module","Apache::Constants")->cpan_file
```

Или, если вы хотите запланировать выполнение задания *cron* для просмотра CPAN, вы можете составить список всех модулей, которые нуждаются в обновлении. Сначала быстрый и грязный способ:

```
perl -e 'use CPAN; CPAN::Shell->r;'
```

Если вам не нужны какие-либо выходные данные, если все модули должны быть актуальными, проанализируйте выходные данные приведенной выше команды для регулярного выражения `/modules are up to date/` и решите отправить выходные данные по почте, только если они не соответствуют.

Если вы предпочитаете делать это в программном стиле в рамках одного процесса, вам может больше подойти что-то вроде этого:

```
# list all modules on my disk that have newer versions on CPAN
for $mod (CPAN::Shell->expand("Module","/./")) {
  next unless $mod->inst_file;
  next if $mod->uptodate;
  printf "Module %s is installed as %s, could be updated to %s from CPAN\n",
      $mod->id, $mod->inst_version, $mod->cpan_version;
}
```

Если это дает слишком много выходных данных каждый день, возможно, вам захочется просмотреть только три модуля. Вы можете написать

```
for $mod (CPAN::Shell->expand("Module","/Apache|LWP|CGI/")) {
```

вместо этого используйте первую строку. Или вы можете комбинировать некоторые из вышеперечисленных приемов:

```
# watch only for a new mod_perl module
$mod = CPAN::Shell->expand("Module","mod_perl");
exit if $mod->uptodate;
# new mod_perl arrived, let me know all update recommendations
CPAN::Shell->r;
```

Методы в других классах

CPAN::Автор::as_glimpse()

Возвращает однострочное описание автора

CPAN::Автор::as_string()

Возвращает многострочное описание автора

CPAN::Автор::email()

Возвращает адрес электронной почты автора

CPAN::Автор::полное имя()

Возвращает имя автора

CPAN::Автор::название()

Псевдоним для полного имени

CPAN::Bundle::as_glimpse()

Возвращает однострочное описание пакета

CPAN::Bundle::as_string()

Возвращает многострочное описание пакета

CPAN::Bundle::clean()

Рекурсивно выполняет метод `clean` для всех элементов, содержащихся в пакете.

CPAN::Bundle::содержит()

Возвращает список идентификаторов объектов, содержащихся в пакете. Связанные объекты могут быть пакетами, модулями или дистрибутивами.

CPAN::Bundle::force($method,@args)

Заставляет CPAN выполнять задачу, которую он обычно отказался бы выполнять. Force принимает в качестве аргументов имя вызываемого метода и любое количество дополнительных аргументов, которые должны быть переданы вызываемому методу. Внутренние компоненты объекта получают необходимые изменения, так что CPAN.pm не отказывается выполнять действие. `force` рекурсивно передается всем содержащимся объектам. Смотрите также раздел выше о `force` и `fforce` прагме.

CPAN::Bundle::get()

Рекурсивно выполняет метод `get` для всех элементов, содержащихся в пакете

CPAN::Bundle::inst_file()

Возвращает самую высокую установленную версию пакета в @INC или `$CPAN::Config->{cpan_home}` . Обратите внимание, что это отличается от CPAN::Module::inst_file.

CPAN::Bundle::inst_version()

Похоже на CPAN::Bundle::inst_file, но возвращает $VERSION

CPAN::Bundle::uptodate()

Возвращает 1, если сам пакет и все его элементы обновлены.

CPAN::Bundle::install()

Рекурсивно выполняет метод `install` для всех элементов, содержащихся в пакете

CPAN::Bundle::make()

Рекурсивно выполняет метод `make` для всех элементов, содержащихся в пакете

CPAN::Bundle::readme()

Рекурсивно выполняет метод `readme` для всех элементов, содержащихся в пакете

CPAN::Bundle::test()

Рекурсивно выполняет метод `test` для всех элементов, содержащихся в пакете

CPAN::Дистрибутив::as_glimpse()

Возвращает однострочное описание дистрибутива

CPAN::Дистрибутив::as_string()

Возвращает многострочное описание дистрибутива

CPAN::Дистрибутив::автор

Возвращает объект CPAN::Author разработчика, загрузившего этот дистрибутив

CPAN::Дистрибутив::pretty_id()

Возвращает строку вида "AUTHORID /TARBALL", где AUTHORID - идентификатор ПАУЗЫ автора, а TARBALL - имя файла рассылки.

CPAN::Дистрибутив::base_id()

Returns the distribution filename without any archive suffix. E.g "Foo-Bar-0.01"

CPAN::Distribution::clean()

Изменяет каталог, из которого был распакован дистрибутив, и запускается `make clean` там.

CPAN::Дистрибутив::содержит модули ()

Возвращает список идентификаторов модулей, содержащихся в файле дистрибутива. Работает только для дистрибутивов, перечисленных в файле 02packages.details.txt.gz. Обычно это означает, что распространяется только самая последняя версия дистрибутива.

CPAN::Дистрибутив::cvs_import()

Изменяет каталог, из которого был распакован дистрибутив, и запускает что-то вроде

```
cvs -d $cvs_root import -m $cvs_log $cvs_dir $userid v$version
```

есть.

CPAN::Дистрибутив::dir()

Возвращает каталог, в который был распакован этот дистрибутив.

CPAN::Распространение::принудительно ($method,@args)

Заставляет CPAN выполнять задачу, которую он обычно отказался бы выполнять. Force принимает в качестве аргументов имя вызываемого метода и любое количество дополнительных аргументов, которые должны быть переданы вызываемому методу. Внутренние компоненты объекта получают необходимые изменения, так что CPAN.pm не отказывается выполнять действие. Смотрите также раздел выше о `force` и `fforce` прагме.

CPAN::Дистрибутив::get()

Загружает дистрибутив с CPAN и распаковывает его. Ничего не делает, если дистрибутив уже был загружен и распакован в течение текущего сеанса.

CPAN::Дистрибутив::установить()

Изменяет каталог, в который был распакован дистрибутив, и запускает там внешнюю команду `make install` . Если `make` еще не была запущена, она будет запущена первой. А `make test` выдается в любом случае, и если это не удается, установка отменяется. Отмены можно избежать, разрешив `force` запустить `install` за вас.

Этот метод установки позволяет установить дистрибутив только в том случае, если на пути нет зависимостей. Чтобы установить объект вместе со всеми его зависимостями, используйте CPAN:: Shell-> установить.

Обратите внимание, что install() не дает значимого возвращаемого значения. Смотрите uptodate() .

CPAN::Дистрибутив::isa_perl()

Возвращает 1, если этот файл дистрибутива кажется дистрибутивом perl. Обычно это выводится только из имени файла, но индекс из CPAN может содержать подсказку для получения возвращаемого значения true и для других имен файлов.

CPAN::Дистрибутив::look()

Изменяет каталог, в который был распакован дистрибутив, и открывает там подоболочку. При выходе из подоболочки возвращается результат.

CPAN::Дистрибутив::make()

Сначала запускается `get` метод, чтобы убедиться, что дистрибутив загружен и распакован. Изменяет каталог, в который был распакован дистрибутив, и запускает там внешние команды `perl Makefile.PL` или `perl Build.PL` и `make`.

CPAN::Дистрибутив::perldoc()

Загружает документацию pod файла, связанного с дистрибутивом (в формате HTML), и запускает его с помощью внешней команды *lynx*, указанной в `$CPAN::Config->{lynx}`. Если *lynx* недоступен, он преобразует его в обычный текст с помощью внешней команды *html2text* и запускает через пейджер, указанный в `$CPAN::Config->{pager}`.

CPAN::Дистрибутив::prefs()

Возвращает ссылку на хэш из первого соответствующего файла YAML, который пользователь поместил в `prefs_dir/` каталог. Выигрывает первое последующее совпадение. Файлы в `prefs_dir/` обрабатываются в алфавитном порядке, а каноническое имя дистрибутива (например, AUTHOR/Foo-Bar-3.14.tar.gz) сопоставляется с регулярными выражениями, хранящимися в значении атрибута $root->{match}{distribution}. Кроме того, все имена модулей, содержащиеся в дистрибутиве, сопоставляются с регулярными выражениями в значении атрибута $root->{match}{module}. Два значения соответствия объединяются. Каждый из двух атрибутов необязателен.

CPAN::Дистрибутив::prereq_pm()

Возвращает ссылку на хэш, которая была объявлена дистрибутивом в качестве элементов `requires` и `build_requires`. Они могут быть объявлены либо `META.yml` (если они являются авторитетными), либо могут быть депонированы после выполнения `Build.PL` в файле `./_build/prereqs` или после выполнения `Makfile.PL`, записанного как `PREREQ_PM` хэш в комментарии к созданному `Makefile`. *Примечание*: этот метод работает только после попытки `make` обновления дистрибутива. В противном случае возвращает undef.

CPAN::Дистрибутив::readme()

Загружает файл README, связанный с дистрибутивом, и запускает его через пейджер, указанный в `$CPAN::Config->{pager}`.

CPAN::Распространение::отчеты()

Загружает данные отчета для этого дистрибутива из www.cpantesters.org и отображает их подмножество.

CPAN::Дистрибутив::read_yaml()

Возвращает содержимое META.yml этого дистрибутива в виде хэш-ссылки. Примечание: работает только после того, как была предпринята попытка `make` запустить дистрибутив. В противном случае возвращает undef. Также возвращает undef, если содержимое META.yml не является авторитетным. (Правила о том, что именно делает содержимое авторитетным, все еще меняются.)

CPAN::Дистрибутив::тест ()

Изменяет каталог, из которого был распакован дистрибутив, и запускается `make test` там.

CPAN::Дистрибутив:: обновление ()

Возвращает 1, если все модули, содержащиеся в дистрибутиве, обновлены. Зависит от containsmods.

CPAN::Index::force_reload()

Принудительно загружает все индексы.

CPAN::Index::reload()

Перезагружает все индексы, если они не считывались более `$CPAN::Config->{index_expire}` дней.

CPAN::InfoObj::dump()

CPAN::Author, CPAN::Bundle, CPAN::Module и CPAN::Distribution наследуют этот метод. Он выводит структуру данных, связанную с объектом. Полезно для отладки. Примечание: структура данных считается внутренней и, следовательно, может быть изменена без предварительного уведомления.

CPAN::Модуль::as_glimpse()

Возвращает однострочное описание модуля в четырех столбцах: Первый столбец содержит слово `Module`, второй столбец состоит из одного символа: знака равенства, если этот модуль уже установлен и актуален, знака меньше, чем, если этот модуль установлен, но может быть обновлен, и пробела, если модуль не установлен. Третий столбец - это название модуля, а четвертый столбец содержит информацию о сопровождающем или распространителе.

CPAN::Module::as_string()

Возвращает многострочное описание модуля

CPAN::Модуль::очистить()

Выполняет очистку дистрибутива, связанного с этим модулем.

CPAN::Модуль::cpan_file()

Возвращает имя файла в CPAN, связанное с модулем.

CPAN::Модуль::cpan_version()

Возвращает последнюю версию этого модуля, доступную на CPAN.

CPAN::Модуль::cvs_import()

Запускает cvs_import в дистрибутиве, связанном с этим модулем.

CPAN::Модуль::описание()

Возвращает описание этого модуля из 44 символов. Доступно только для модулей, перечисленных в списке модулей (CPAN/modules/00modlist.long.html или 00modlist.long.txt.gz)

CPAN::Модуль::дистрибутив ()

Возвращает объект CPAN::Distribution, содержащий текущую версию этого модуля.

CPAN::Модуль::dslip_status()

Возвращает ссылку на хэш. Ключами хэша являются буквы D , S , L , I и <P>, обозначающие статус разработки, уровень поддержки, язык, интерфейс и общедоступную лицензию соответственно. Данные о состоянии DSLIP собираются pause.perl.org когда авторы регистрируют свои пространства имен. Значения 5 хэш-элементов представляют собой односимвольные слова, значение которых описано в таблице ниже. Также есть 5 хэш-элементов DV , SV , LV , IV и <PV>, которые содержат более подробное значение 5 переменных состояния.

Где символы 'DSLIP' имеют следующие значения:

```
D - Development Stage  (Note: *NO IMPLIED TIMESCALES*):
  i   - Idea, listed to gain consensus or as a placeholder
  c   - under construction but pre-alpha (not yet released)
  a/b - Alpha/Beta testing
  R   - Released
  M   - Mature (no rigorous definition)
  S   - Standard, supplied with Perl 5

S - Support Level:
  m   - Mailing-list
  d   - Developer
  u   - Usenet newsgroup comp.lang.perl.modules
  n   - None known, try comp.lang.perl.modules
  a   - abandoned; volunteers welcome to take over maintenance

L - Language Used:
  p   - Perl-only, no compiler needed, should be platform independent
  c   - C and perl, a C compiler will be needed
  h   - Hybrid, written in perl with optional C code, no compiler needed
  +   - C++ and perl, a C++ compiler will be needed
  o   - perl and another language other than C or C++

I - Interface Style
  f   - plain Functions, no references used
  h   - hybrid, object and function interfaces available
  n   - no interface at all (huh?)
  r   - some use of unblessed References or ties
  O   - Object oriented using blessed references and/or inheritance

P - Public License
  p   - Standard-Perl: user may choose between GPL and Artistic
  g   - GPL: GNU General Public License
  l   - LGPL: "GNU Lesser General Public License" (previously known as
        "GNU Library General Public License")
  b   - BSD: The BSD License
  a   - Artistic license alone
  2   - Artistic license 2.0 or later
  o   - open source: approved by www.opensource.org
  d   - allows distribution without restrictions
  r   - restricted distribution
  n   - no license at all
```

CPAN::Module::force($method,@args)

Заставляет CPAN выполнять задачу, которую он обычно отказывается выполнять. Force принимает в качестве аргументов имя вызываемого метода и любое количество дополнительных аргументов для передачи этого метода. Внутренние компоненты объекта получают необходимые изменения, так что CPAN.pm не отказывается выполнять действие. Смотрите также раздел выше о force и fforce прагме.

CPAN::Модуль::get()

Запускает get для дистрибутива, связанного с этим модулем.

CPAN::Module::inst_file()

Возвращает имя файла модуля, найденного в @INC. Сообщается о первом найденном файле, точно так же, как сам perl прекращает поиск в @INC, как только находит модуль.

CPAN::Модуль::доступный файл()

Возвращает имя файла модуля, найденного в PERL5LIB или @INC. Сообщается о первом найденном файле. Преимущество этого метода перед `inst_file` заключается в том, что модули, которые были протестированы, но еще не установлены, включены, поскольку PERL5LIB отслеживает протестированные модули.

CPAN::Module::inst_version()

Возвращает номер версии установленного модуля в удобочитаемом формате.

CPAN::Module::available_version()

Возвращает номер версии доступного модуля в удобочитаемом формате.

CPAN::Module::install()

Запускает `install` дистрибутив, связанный с этим модулем.

CPAN::Module::look()

Изменяет каталог, в который был распакован дистрибутив, связанный с этим модулем, и открывает там подоболочку. Возвращается выход из подоболочки.

CPAN::Модуль::make()

Работает `make` в дистрибутиве, связанном с этим модулем.

CPAN::Module::manpage_headline()

Если модуль установлен, просматривает справочную страницу модуля, считывает заголовок и возвращает его. Более того, если модуль был загружен в течение этого сеанса, выполняет аналогичное действие с загруженным модулем, даже если он еще не был установлен.

CPAN::Модуль::perldoc()

Запускает `perldoc` на этом модуле.

CPAN::Модуль::readme()

Работает `readme` в дистрибутиве, связанном с этим модулем.

CPAN::Модуль::отчеты()

Вызывает метод reports() для связанного объекта распространения.

CPAN::Модуль::тест()

Работает `test` в дистрибутиве, связанном с этим модулем.

CPAN::Module::uptodate()

Возвращает 1, если модуль установлен и обновлен.

CPAN::Module::userid()

Возвращает идентификатор автора модуля.

Диспетчер кэша

В настоящее время диспетчер кэша отслеживает только каталог сборки ($CPAN::Config->{build_dir}). Это простой механизм FIFO, который удаляет полные каталоги ниже, `build_dir` как только размер всех каталогов там становится больше, чем $ CPAN::Config->{build_cache} (в МБ). Содержимое этого кэша может быть использовано для последующих переустановок, которые вы собираетесь выполнить вручную, но сам CPAN никогда не будет доверять ему. Это связано с тем, что пользователь может использовать эти каталоги для создания модулей на разных архитектурах.

Существует другой каталог ($CPAN::Config->{keep_source_where}), в котором хранятся исходные файлы дистрибутива. Этот каталог не покрывается менеджером кэша и должен управляться пользователем. Если вы выберете тот же каталог, что и build_dir и как каталог keep_source_where , то ваши исходники будут удалены с помощью того же механизма fifo.

Пакеты

Пакет - это просто модуль perl в пространстве имен Bundle::, который не определяет никаких функций или методов. Обычно он содержит только документацию.

Он запускается как модуль perl с объявлением пакета и переменной $VERSION. После этого раздел pod выглядит как любой другой pod, с той лишь разницей, что *существует один специальный раздел pod*, начинающийся с (дословно):

```
=head1 CONTENTS
```

В этом разделе модуля каждая строка соответствует формату

```
Module_Name [Version_String] [- optional text]
```

Единственная обязательная часть - это первое поле, имя модуля (например, Foo::Bar, т.е. *Не* имя файла дистрибутива). Остальная часть строки необязательна. Часть комментариев разделена тире, как и в заголовке справочной страницы.

Распространение пакета должно соответствовать тем же правилам, что и в других дистрибутивах.

Пакеты обрабатываются особым образом в пакете CPAN. Если вы скажете "установить пакет:: Tkkit" (при условии, что такой пакет существует), CPAN установит все модули в разделе СОДЕРЖИМОГО модуля. Вы можете установить свои собственные пакеты локально, поместив соответствующий файл пакета где-нибудь в вашем @INC пути. Команда autobundle(), доступная в интерфейсе командной оболочки, делает это за вас, включая все установленные в данный момент модули в файл snapshot bundle.

ПРЕДВАРИТЕЛЬНЫЕ ТРЕБОВАНИЯ

Программа CPAN старается зависеть от как можно меньшего количества ресурсов, чтобы пользователь мог использовать ее во враждебной среде. Она работает тем лучше, чем больше полезных функций предоставляет среда. Например, если вы попробуете в оболочке CPAN

```
install Bundle::CPAN
```

или

```
install Bundle::CPANxxl
```

вы обнаружите, что оболочка более удобна, чем обычная оболочка раньше.

Если у вас есть локальное зеркало CPAN и вы можете получить доступ ко всем файлам с URL-адресами "file:", то для запуска этого модуля вам нужен только perl более поздней версии, чем perl5.003. В противном случае настоятельно рекомендуется использовать Net:: FTP. LWP может потребоваться для систем, отличных от UNIX, или если ваш ближайший сайт CPAN связан с URL, который таковым не является `ftp:` .

Если у вас нет ни Net:: FTP, ни LWP, реализован запасной механизм для внешней команды ftp или для внешней команды lynx.

УТИЛИТЫ

Поиск пакетов и ВЕРСИИ

Этот модуль предполагает, что все пакеты на CPAN

- объявляйте их переменную $VERSION простым для анализа способом. Это предварительное условие вряд ли можно ослабить, поскольку оно потребляет слишком много памяти для загрузки всех пакетов в запущенную программу только для определения переменной $VERSION. В настоящее время все программы, работающие с версией, используют что-то вроде этого

  ```
  perl -MExtUtils::MakeMaker -le \
      'print MM->parse_version(shift)' filename
  ```

  Если вы являетесь автором пакета и задаетесь вопросом, можно ли проанализировать вашу версию в $ VERSION, пожалуйста, попробуйте описанный выше метод.
- поставляются в виде сжатых или gzipped tarfiles или в виде zip-файлов и содержат `Makefile.PL` или `Build.PL` (ну, мы стараемся обрабатывать немного больше, но без особого энтузиазма).

Отладка

Отладка этого модуля более чем сложна из-за вмешательства программного обеспечения, создающего индексы на CPAN, процесса зеркального отображения на CPAN, упаковки, конфигурации, синхронности и даже (ого!) из-за ошибок в CPAN.pm сам модуль.

Для отладки самого кода CPAN.pm в интерактивном режиме можно включить некоторые средства отладки для большинства пакетов внутри CPAN.pm с помощью одного из

o пакет отладки...

устанавливает режим отладки для пакетов.

o отладочный пакет...

отключает режим отладки для пакетов.

o отлаживать все

включает отладку для всех пакетов.

o номер отладки

который устанавливает пакеты отладки напрямую. Обратите внимание, что `o debug 0` отключает отладку.

Что кажется успешной стратегией, так это сочетание `reload cpan` и переключателей отладки. Добавьте новую инструкцию debug во время выполнения в командной оболочке, а затем выполните a `reload cpan` и сразу увидите новые сообщения отладки без потери текущего контекста.

`o debug` без указания аргумента выводит список допустимых имен пакетов и текущего набора пакетов в режиме отладки. `o debug` имеет встроенную поддержку завершения.

Для отладки данных CPAN существует `dump` команда, которая принимает те же аргументы, что и make/test/install, и выводит данные каждого объекта::Dumper dump. Если аргумент выглядит как переменная perl и содержит одно из `$` , `@` или `%` , он вычисляется () и передается непосредственно в Data::Dumper .

Гибкий файл, Zip, автономный режим

CPAN.pm прекрасно работает и без доступа к сети. Если вы обслуживаете компьютеры, которые вообще не подключены к сети, вам следует подумать о работе с `file:` URL-адресами. Сначала вам придется где-то собрать свои модули. Поэтому вы можете использовать CPAN.pm для объединения всего необходимого на компьютере, подключенном к сети. Затем скопируйте каталог $CPAN::Config->{keep_source_where} (но не $CPAN::Config->{build_dir}) на дискету. Эта дискета является своего рода персональной CPAN. CPAN.pm на компьютерах, не подключенных к сети, эта дискета прекрасно работает. Смотрите также ниже параграф о поддержке CD-ROM.

Базовые утилиты для программистов

has_inst ($module)

Возвращает true, если модуль установлен. Используется для загрузки всех модулей в запущенную систему CPAN.pm которые считаются необязательными. Переменная config `dontload_list` перехватывает `has_inst()` вызов таким образом, что дополнительный модуль не загружается, несмотря на то, что он доступен. Например, следующая команда предотвратит `YAML.pm` загрузку:

```
cpan> o conf dontload_list push YAML
```

Смотрите источник для получения подробной информации.

use_inst ($module)

Аналогично has_inst() пытается загрузить дополнительную библиотеку, но также умирает, если библиотека недоступна

has_usable($module)

Возвращает true, если модуль установлен и находится в рабочем состоянии. Полезно только для нескольких модулей, которые используются внутри компании. Подробности смотрите в источнике.

экземпляр ($module)

Конструктор для всех одиночных элементов, используемых для представления модулей, дистрибутивов, авторов и пакетов. Если объект уже существует, этот метод возвращает объект; в противном случае он вызывает конструктор.

интерфейс ()

интерфейс ($new_frontend)

Средство получения / установки для объекта внешнего интерфейса. Метод просто позволяет создавать подклассы CPAN.pm.

БЕЗОПАСНОСТЬ

В нем нет надежного уровня безопасности CPAN.pm. CPAN.pm помогает установить на ваш компьютер чужой, незамаскированный, неподписанный код. Мы сравниваем с контрольной суммой, которая поступает из сети так же, как и сам файл рассылки. Но мы стараемся упростить добавление безопасности по требованию:

Модули с криптографической подписью

Начиная с версии 1.77, CPAN.pm появилась возможность проверять дистрибутивы модулей с криптографической подписью с помощью Module::Signature . Модули CPAN могут быть подписаны их авторами, что обеспечивает большую безопасность. Простые контрольные суммы MD5 без знака, которые ранее использовались CPAN, защищают в основном от случайного повреждения файлов.

Вам необходимо установить Module::Signature, что, в свою очередь, требует наличия хотя бы одного из модулей Crypt:: OpenPGP или инструмента *gpg* командной строки.

Вам также потребуется иметь возможность подключаться через Интернет к серверам открытых ключей, таким как pgp.mit.edu, и их порту 11731 (протокол HKP).

Параметр конфигурации check_sigs предназначен для включения или выключения проверки подписи.

ЭКСПОРТ

Большинство функций в пакете CPAN экспортируются по умолчанию. Причина этого в том, что основное использование предназначено для оболочки cpan или для однострочников.

СРЕДА

Когда оболочка CPAN входит в подоболочку с помощью команды look , она устанавливает для среды CPAN_SHELL_LEVEL значение 1 или увеличивает эту переменную, если она уже установлена.

Когда CPAN запускается, он присваивает переменной окружения PERL5_CPAN_IS_RUNNING идентификатор запущенного процесса. Он также устанавливает PERL5_CPANPLUS_IS_RUNNING для предотвращения неконтролируемых процессов, которые могли произойти со старыми версиями Module::Install.

При запуске perl Makefile.PL переменной среды PERL5_CPAN_IS_EXECUTING устанавливается полный путь к Makefile.PL выполняемому модулю. Это предотвращает неконтролируемые процессы с более новыми версиями Module::Install .

Когда установлена конфигурационная переменная ftp_passive, все загрузки будут выполняться с переменной окружения FTP_PASSIVE, имеющей это значение. В целом это хорошая идея, поскольку она влияет как на подключения на основе Net :: FTP, так и на основе LWP. Того же эффекта можно добиться, запустив оболочку cpan с этим набором переменных окружения. Только для Net :: FTP также всегда можно установить пассивный режим, запустив libnetcfg.

ЗАПОЛНИТЬ УСТАНОВКУ БОЛЬШИМ КОЛИЧЕСТВОМ МОДУЛЕЙ

Заполнить свежеустановленный perl вашими любимыми модулями довольно просто, если вы поддерживаете частный файл определения пакета. Чтобы получить полезную схему файла определения пакета, в командной строке оболочки CPAN можно использовать команду autobundle. Эта команда записывает файл определения пакета для всех модулей, установленных для текущего интерпретатора perl. Рекомендуется запускать эту команду только один раз, и с этого момента поддерживать файл вручную под личным именем, скажем, Bundle /my_bundle.pm . С помощью умного файла bundle вы можете просто сказать

```
cpan> install Bundle::my_bundle
```

затем ответьте на несколько вопросов и сходите выпить кофе (возможно, даже в другом городе).

Ведение файла определения пакета означает отслеживание двух вещей: зависимостей и интерактивности. CPAN.pm иногда при вычислении зависимостей происходит сбой, поскольку не все модули правильно определяют все атрибуты MakeMaker, поэтому в файле определения пакета следует указать предварительные требования как можно раньше. С другой стороны, раздражает, что так много дистрибутивов нуждаются в некоторой интерактивной настройке. Итак, что вы можете попытаться сделать в своем файле private bundle, так это поместить пакеты, которые необходимо настроить на ранней стадии, в файл, а более щадящие - позже, чтобы через несколько минут вы могли пойти выпить кофе и уйти. CPAN.pm для автоматической обработки.

РАБОТА С CPAN.pm ЗА БРАНДМАУЭРАМИ

Спасибо Грэму Барру за публикацию следующих параграфов о взаимодействии perl с различными конфигурациями брандмауэров. Для получения дополнительной информации о брандмауэрах рекомендуется обратиться к документации, прилагаемой к программе *ncftp*. Если вы не можете пройти через брандмауэр с помощью простой настройки Perl, вероятно, вы можете настроить *ncftp* так, чтобы он работал через ваш брандмауэр.

Три основных типа брандмауэров

Брандмауэры можно разделить на три основных типа.

http firewall

В этом случае брандмауэр запускает веб-сервер, и для доступа к внешнему миру вы должны делать это через этот веб-сервер. Если вы задаете переменным среды, таким как http_proxy или ftp_proxy, значения, начинающиеся с http: //, или в вашем веб-браузере установлена информация о прокси, то вы знаете, что работаете за брандмауэром http.

Для доступа к серверам вне этих типов брандмауэров с помощью perl (даже для ftp) вам нужен LWP или HTTP::Tiny.

брандмауэр ftp

Здесь брандмауэр запускает ftp-сервер. Этот тип брандмауэра позволяет вам получать доступ только к ftp-серверам за пределами брандмауэра. Обычно это делается путем подключения к брандмауэру с помощью ftp, затем ввода имени пользователя типа "user@outside.host.com".

Для доступа к серверам вне брандмауэров этого типа с помощью perl вам нужен Net :: FTP.

Односторонняя видимость

Односторонняя видимость означает, что эти брандмауэры пытаются стать невидимыми для пользователей внутри брандмауэра. FTP-соединение для передачи данных обычно создается путем отправки вашего IP-адреса на удаленный сервер и последующего прослушивания обратного соединения. Но удаленный сервер не сможет подключиться к вам из-за брандмауэра. Для этих типов брандмауэров FTP-соединения должны выполняться в пассивном режиме.

Есть два варианта, которые я могу придумать.

SOCKS

Если вы используете брандмауэр SOCKS, вам нужно будет скомпилировать perl и связать его с библиотекой SOCKS. Это то, что обычно называется perl с socksified. С помощью этого исполняемого файла вы сможете подключаться к серверам за пределами брандмауэра, как если бы его там не было.

IP Masquerade

Это когда брандмауэр, реализованный в ядре (через NAT, или преобразование сетевых адресов), позволяет скрыть всю сеть за одним IP-адресом. С этим брандмауэром специальная компиляция не требуется, поскольку вы можете обращаться к хостам напрямую.

Для доступа к ftp-серверам за такими брандмауэрами обычно требуется установить переменной окружения `FTP_PASSIVE` или конфигурационной переменной ftp_passive значение true.

Настройка lynx или ncftp для прохождения через брандмауэр

Если вы можете пройти через свой брандмауэр, например, с помощью lynx, предположительно, с помощью такой команды, как

```
/usr/local/bin/lynx -pscott:tiger
```

затем вы должны настроить CPAN.pm с помощью команды

```
o conf lynx "/usr/local/bin/lynx -pscott:tiger"
```

Вот и все. Аналогично для ncftp или ftp вы должны настроить что-то вроде

```
o conf ncftp "/usr/bin/ncftp -f /home/scott/ncftplogin.cfg"
```

Ваш пробег может отличаться...

ЧАСТО задаваемые вопросы

1)

Я установил новую версию модуля X, но CPAN продолжает говорить, что у меня установлена старая версия

Вероятно, у вас **действительно** установлена старая версия. Это может произойти, если модуль установится в другой каталог по пути @INC, чем он был установлен ранее. На самом деле это не проблема CPAN.pm у вас возникнет такая же проблема при установке модуля вручную. Самый простой способ предотвратить такое поведение - добавить аргумент `UNINST=1` в `make install` вызов, и именно поэтому многие люди добавляют этот аргумент постоянно, настраивая

```
o conf make_install_arg UNINST=1
```

2)

Итак, почему UNINST = 1 не используется по умолчанию?

Потому что есть люди, у которых есть свои точные ожидания относительно того, кто где может устанавливать в пути @INC и кто использует какой массив @INC. В точно настроенных средах `UNINST=1` может привести к повреждению.

3)

Я хочу навести порядок и установить новый perl вместе со всеми имеющимися у меня модулями. Как мне это сделать?

Запустите команду autobundle для вашего старого perl и, при необходимости, переименуйте результирующий файл bundle (например, Bundle/mybundle.pm), установите новый perl с префиксом параметра Configure, например

```
./Configure -Dprefix=/usr/local/perl-5.6.78.9
```

Установите файл пакета, который вы создали на первом шаге, с чем-то вроде

```
cpan> install Bundle::mybundle
```

и все готово.

4)

Когда я устанавливаю пакеты или несколько модулей одной командой, выводится слишком много данных, за которыми невозможно уследить.

Возможно, вам захочется настроить что-то вроде

```
o conf make_arg "| tee -ai /root/.cpan/logs/make.out"
o conf make_install_arg "| tee -ai /root/.cpan/logs/make_install.out"
```

чтобы стандартный вывод был записан в файл для последующей проверки.

5)

Я не root, как я могу установить модуль в личный каталог?

Начиная с версии CPAN 1.9463, если у вас нет разрешения на запись каталогов библиотеки perl по умолчанию, процесс настройки CPAN спросит вас, хотите ли вы использовать bootstrap <local::lib>, что упрощает ведение личного каталога библиотеки perl.

Еще одна вещь, которую вам следует иметь в виду, это то, что параметр UNINST может быть опасен при установке в личный кабинет, потому что вы можете случайно удалить модули, от которых зависят другие люди, которые не используют личный кабинет.

6)

Как получить пакет, развернуть его и внести изменения перед его сборкой?

Взгляните на `look (!)` команду.

7)

Я установил пакет и столкнулся с парой сбоев. Когда я повторил попытку, все разрешилось нормально. Можно ли это исправить, чтобы работало с первой попытки?

Причина этого в том, что CPAN не знает зависимостей всех модулей при запуске. Чтобы выбрать дополнительные элементы для установки, он просто использует данные, найденные в файле META.yml или сгенерированном Makefile. Необнаруженная недостающая часть прерывает процесс. Но вполне может быть, что ваш пакет устанавливает какое-то предварительное условие позже, чем какой-то зависимый элемент, и, таким образом, ваша вторая попытка может решить все. Пожалуйста, обратите внимание, CPAN.pm не знает дерево зависимостей заранее и не может отсортировать очередь объектов для установки в топологически правильном порядке. Это отлично решается, **если** все модули правильно объявляют предварительные требования с помощью атрибута PREREQ_PM для MakeMaker или `requires` строки Module::Build. Для пакетов, которые выходят из строя и которые необходимо часто устанавливать, рекомендуется отсортировать файл определения пакета вручную.

8)

В нашей внутренней сети у нас есть много модулей для внутреннего использования. Как я могу интегрировать эти модули с CPAN.pm, но без загрузки модулей в CPAN?

Взгляните на модуль CPAN::Site.

9)

Когда я запускаю оболочку CPAN, я получаю сообщение об ошибке в моем `/etc/inputrc` (или `~/.inputrc`) файле.

Это проблемы с readline, которые можно устранить, только изучив конфигурацию readline в вашей архитектуре и соответствующим образом изменив файл, на который даны ссылки. Пожалуйста, сделайте резервную копию `/etc/inputrc` или `~/.inputrc` и отредактируйте их. Довольно часто проблему решают безобидные изменения, такие как перевод некоторых аргументов в верхний или нижний регистр.

10)

У некоторых авторов в именах странные символы.

Внутренне CPAN.pm использует кодировку UTF-8. Если ваш терминал ожидает кодировку ISO-8859-1, конвертер можно активировать, установив для term_is_latin значение true в вашем конфигурационном файле. Одним из способов сделать это было бы

```
cpan> o conf term_is_latin 1
```

Если требуется поддержка другой кодировки, пожалуйста, отправьте сообщение об ошибке по адресу CPAN.pm по адресу rt.cpan.org и опишите свои потребности. Возможно, мы сможем расширить поддержку или, возможно, терминалы UTF-8 станут широко доступными.

Примечание: эта конфигурационная переменная устарела и будет удалена в будущей версии CPAN.pm. Она будет заменена соглашениями, касающимися семейства переменных окружения $ LANG и $ LC_ *.

11)

Когда по какой-либо причине установка завершается неудачей, а затем я исправляю условие ошибки и повторяю попытку, CPAN.pm отказывается устанавливать модуль, говоря `Already tried without success`.

Используйте force pragma следующим образом

```
force install Foo::Bar
```

Или вы можете использовать

```
look Foo::Bar
```

а затем `make install` непосредственно в подоболочке.

12)

Как мне установить "ВЕРСИЮ модуля для РАЗРАБОТЧИКОВ"?

По умолчанию CPAN устанавливает последнюю версию модуля, выпущенную не разработчиком. Если вы хотите установить версию разработчика, вы должны указать частичный путь, начинающийся с идентификатора автора, к архиву, который вы хотите установить, например:

```
cpan> install KWILLIAMS/Module-Build-0.27_07.tar.gz
```

Обратите внимание, что вы можете использовать команду `ls` , чтобы указать этот путь.

13)

Как мне установить модуль и все его зависимости из командной строки без каких-либо запросов, несмотря на мою конфигурацию CPAN (или ее отсутствие)?

CPAN использует функцию ExtUtils::MakeMaker prompt(), чтобы задавать свои вопросы, поэтому, если вы установите переменную среды PERL_MM_USE_DEFAULT , вам вообще не должны задавать никаких вопросов (при условии, что устанавливаемые вами модули также хорошо подчиняются этой переменной):

```
% PERL_MM_USE_DEFAULT=1 perl -MCPAN -e 'install My::Module'
```

14)

Как мне создать сборку на основе модуля::Build.PL, производный от ExtUtils ::ориентированный на MakeMaker Makefile.PL ?

http://search.cpan.org/dist/Module-Build-Convert/

15)

Меня часто раздражает неспособность CPAN shell помочь мне выбрать хорошее зеркало.

CPAN теперь может помочь вам выбрать "хорошее" зеркало, основываясь на том, какие из них имеют наименьшее время "пинга" в оба конца. Из командной оболочки используйте команду 'o conf init urllist' и разрешите CPAN автоматически выбирать зеркала для вас.

Помимо этой справки, вам доступен параметр конфигурации urllist. Вы можете добавлять и удалять сайты по своему усмотрению. Вам следует выяснить, какие сайты обладают наилучшей актуальностью, пропускной способностью, надежностью и т.д. И топологически близки к вам. Одни люди предпочитают быструю загрузку, другие - актуальность, третьи - надежность. Вы сами решаете, какие из них попробовать в каком порядке.

Хенк П. Пеннинг поддерживает сайт, который собирает данные о сайтах CPAN:

```
http://mirrors.cpan.org/
```

Кроме того, не стесняйтесь поиграть с экспериментальными функциями. Запустите

```
o conf init randomize_urllist ftpstats_period ftpstats_size
```

и выберите ваши любимые параметры. После нескольких загрузок выполнение `hosts` команды, вероятно, поможет вам выбрать лучшие зеркальные сайты.

16)

Почему мне задают одни и те же вопросы каждый раз, когда я запускаю оболочку?

Вы можете сделать изменения конфигурации постоянными, вызвав команду `o conf commit` . В качестве альтернативы установите для `auto_commit` переменной значение true, выполнив команду `o conf init auto_commit` и ответив утвердительно на следующий вопрос.

17)

В старых версиях CPAN.pm исходный корневой каталог всех архивных файлов находился в каталоге сборки. Теперь к именам этих каталогов всегда добавляются случайные символы. Зачем это было сделано?

Случайные символы предоставлены File::Temp и гарантируют, что отдельный каталог сборки каждого модуля уникален. Это делает запуск CPAN.pm одновременно в параллельных процессах безопасным.

18)

Кстати, о каталоге сборки. Должен ли я сам его очищать?

У вас есть выбор установить для переменной config scan_cache значение never . Затем вы должны очистить ее самостоятельно. Другие возможные значения, atstart и atexit очищают каталог сборки при запуске (или, точнее, после первого извлечения в каталог сборки) или при выходе из оболочки CPAN, соответственно. Если вы никогда не запускаете оболочку CPAN, вам, вероятно, также придется самостоятельно очищать каталог сборки.

19)

Как я могу переключиться на sudo вместо local::lib?

Следующие 5 проверяемых параметров среды необходимо сбросить до предыдущих значений: PATH, PERL5LIB, PERL_LOCAL_LIB_ROOT, PERL_MB_OPT, PERL_MM_OPT; и эти два CPAN.pm необходимо перенастроить переменные конфигурации: make_install_make_command и mbuild_install_build_command. Пять переменных env, вероятно, были перезаписаны в вашем $HOME /.bashrc или каком-либо эквиваленте. Вы либо находите их там и удаляете их трассировки и выходите из системы, либо временно переопределяете их, в зависимости от вашего конкретного желания. Две переменные конфигурации cpanpm могут быть установлены с помощью:

```
o conf init /install_.*_command/
```

вероятно, за этим последует

```
o conf commit
```

СОВМЕСТИМОСТЬ

СТАРЫЕ ВЕРСИИ PERL

CPAN.pm регулярно тестируется для работы под 5.005 и различными более новыми версиями. Становится все сложнее обеспечить выполнение минимальных предварительных условий для работы со старыми perls. Заставить работать весь пакет ::CPAN практически невозможно. Если вы можете использовать только эти старые версии, имейте в виду, что CPAN рассчитан на нормальную работу без установленного пакета :: CPAN.

Для начала обратите внимание, что GBARR/Scalar-List-Utils-1.18.tar.gz совместим с ancient perls и что File::Temp указан в качестве обязательного условия, но CPAN имеет разумные обходные пути, если он отсутствует.

CPANPLUS

Этот модуль и его конкурент, модуль CPANPLUS, оба намного круче другого. CPAN.pm старше. CPANPLUS был разработан как более модульный, но он никогда не предполагался совместимым с CPAN.pm.

CPANMINUS

В 2010 году было запущено приложение :: cpanminus как новый подход к оболочке cpan со значительно меньшими размерами. Очень классная штука.

РЕКОМЕНДАЦИИ ПО БЕЗОПАСНОСТИ

Это программное обеспечение позволяет обновлять программное обеспечение на вашем компьютере и, следовательно, по своей сути опасно, поскольку недавно установленное программное обеспечение может содержать ошибки и изменять работу вашего компьютера или даже делать его непригодным для использования. Пожалуйста, подумайте о создании резервной копии ваших данных перед каждым обновлением.

ОШИБКИ

Пожалуйста, сообщайте об ошибках через http://rt.cpan.org /

Перед отправкой сообщения об ошибке, пожалуйста, убедитесь, что традиционный метод сборки пакета модулей Perl из командной строки, следуя инструкциям по установке этого пакета, все еще работает в вашей среде.

АВТОР

Andreas Koenig <andk@cpan.org>

ЛИЦЕНЗИЯ

Эта программа является бесплатным программным обеспечением; вы можете распространять ее и / или изменять на тех же условиях, что и сам Perl.

Посмотреть http://www.perl.com/perl/misc/Artistic.html

ПЕРЕВОДЫ

Каваи, Таканори предоставляет японский перевод очень старой версии этой справочной страницы по адресу http://homepage3.nifty.com/hippo2000/perltips/CPAN.htm

СМОТРЕТЬ ТАКЖЕ

Многие люди входят в оболочку CPAN, запуская служебную программу cpan, которая установлена в том же каталоге, что и сам perl. Итак, если у вас есть этот каталог в вашей переменной PATH (или какой-либо эквивалент в вашей операционной системе), то ввод `cpan` в окне консоли также будет работать для вас. Выше утилита предоставляет несколько сочетаний клавиш командной строки.

мележик (Алексей) прислал мне ссылку, где он опубликовал рецепт шеф-повара для работы CPAN.pm: http://community.opscode.com/cookbooks/cpan.

Браузер Perldoc поддерживается Дэном Буком (DBOOK). Пожалуйста, свяжитесь с ним через GitHub issue tracker или по электронной почте по поводу любых проблем с самим сайтом, поиском или отображением документации.

Документация по Perl поддерживается разработчиками Perl 5 при разработке Perl. Пожалуйста, свяжитесь с ними через Perl issue tracker, список рассылки или IRC, чтобы сообщить о любых проблемах с содержанием или форматом документации.