

Rakudo – компилятор Perl 6 на виртуальной машине Parrot

Андрей Шитов

Rakudo – новое название компилятора Perl 6, входящего в дистрибутив виртуальной машины Parrot. Официальный сайт проекта – www.rakudo.org. И компилятор, и название – свежее направление в развитии шестой версии языка: первая запись в блоге разработчиков на этом сайте появилась 17 января 2008 года.

Название предложил Дамиан Конвей, оно происходит от японского перевода фразы The Way Of The Camel: Rakuda-do. Сокращенный вариант Rakudo (который по-японски к тому же имеет смысл «рай») и стал названием нового проекта.

Появление Rakudo означает, что разработчики вернулись к первоначальной идее создать компилятор Perl 6, который будет переводить программу в некий промежуточный байткод, исполняемый виртуальной машиной.

Установка

Сейчас Rakudo является частью дистрибутива виртуальной машины Parrot и находится в каталоге `languages/perl6`. В будущем разработчики планируют перенести код в отдельный каталог или даже в отдельный репозиторий. Для установки Rakudo нужно загрузить дистрибутив Parrot либо из SVN-репозитория (<https://svn.perl.org/parrot/trunk>), либо по ссылке – <http://svn.perl.org/snapshots/parrot/parrot-latest.tar.gz>, а затем выполнить стандартные команды:

```
perl Configure.pl
make
make install
```

Ожидается, что в системе уже установлен пакет ICU, International Components for Unicode, (с соответствующими библиотеками `libicuuc.so` и `libicudata.so`). Если их нет, об этом нужно сообщить конфигуратору:

```
perl Configure.pl --without-icu
```

После этого будет получена виртуальная машина, но компилятор Perl 6

требуется собрать отдельно: нужно перейти в каталог `languages/perl6` и выполнить там команду `make`:

```
cd languages/perl6/
make perl6
```

Это создаст файл `perl6.pbc` (расширение `pbc` обозначает Parrot Byte Code), который по сути и является компилятором Perl 6.

Запуск программ на Perl 6 выполняется либо явно через Parrot:

```
parrot perl6.pbc programme.pl
```

либо просто:

```
perl6 programme.pl
```

Если не указать имя программы, Rakudo перейдет в интерактивный режим.

Для установки Parrot под Windows намного проще скачать инсталлятор с сайта <http://parrotwin32.sourceforge.net>. В комплекте сразу идет программа `perl6.exe`.

Комплект Rakudo

Реализация языка Perl 6 в Rakudo основана на стандартной грамматике, описанной в файле `STD.pm`, находящемся в репозитории Pugs: <http://svn.pugscode.org/pugs/src/perl6/STD.pm>.

Разработчики стремятся максимально следовать спецификации, описанной в `STD.pm`, и уже охватили значительную ее часть, но в каждом случае выполнена только базовая реализация отдельного фрагмента.

В папке `languages/perl6/t` содержится небольшой набор тестов, и предполагается, что в идеале Rakudo дол-

жен проходить тесты на спецификацию языка, которые сейчас хранятся опять же в репозитории Pugs: <http://svn.pugscode.org/pugs/t>.

Тесты

С одной стороны, Rakudo появился через несколько лет после того, как был создан Pugs, с помощью которого удалось попробовать многие возможности Perl 6, с другой – Rakudo это новый проект, поэтому он проходит свой собственный цикл развития, и многие конструкции языка здесь еще не реализованы. В то же время, благодаря активной работе программистов, новые возможности появляются довольно быстро. Все описанные далее простые программы работают в реализации Rakudo, входящей в Parrot 0.6.0. На предупреждения и ошибки, которые могут возникнуть при выполнении примеров, пока не стоит обращать особого внимания.

Связывание переменных

В Perl 6 существует механизм, называемый `binding`, создающий ссылки на переменные. Синтаксис связывания – `$link := $var`. Например:

```
my $hour = 14;
my $summertime := $hour;
say $hour;
$summertime++;
say $hour;
```

Эта программа дважды выводит на печать переменную `$hour`, но при этом печатает разные значения, хотя явного изменения этой переменной в программе нет.

for

Оператор `for` реализован более или менее полно. Например, уже сейчас

YAPC::Russia

May Perl 2008

Конференция про Perl

17—18 мая 2008, Москва

Смысл *May Perl* —
рассказать и услышать про то,
что *может* и где *может* быть использован Perl.

mail@perlruussia.ru

2008.perlruussia.ru

Приглашаем вас стать участниками,
докладчиками и спонсорами

Точное место проведения
будет объявлено на сайте,
но регистрироваться можно
уже сейчас

Что мы уже провели:

Perl Mova 2008.perlukraine.org Perl Today 2007.perlruussia.ru

Планируется серия
пятиминутных
блиц-докладов:
perl.lv/lt

Генеральный спонсор



Спонсор ПО



Информационный спонсор



Организаторы



можно пользоваться привлекательной конструкцией «for @value -> \$var». К сожалению, выбирать за один шаг более двух значений пока нельзя.

```
my @values = <
    registration
    lunch
    coffee-break
    closing
>;
my $c = 0;
for @values -> $event {
    $c++;
    say "$c. $event";
}
```

Классы, isa и WHAT

В Rakudo реализованы элементы ООП, в частности, присутствует примитивная поддержка синтаксиса классов и сопутствующие методы WHAT и isa.

Вот пример простого класса, который использует закрытую переменную \$!Name. На закрытость указывает вторичный сигил (восклицательный знак).

```
class Language {
    has $!Name;
    method give_name ($newname) {
        $!Name = $newname;
    }
    method say_name {
        say "This is $!Name";
    }
}
my $lang = Language.new();
$lang.give_name('Perl 6');
$lang.say_name();
```

Попытка узнать имя класса через вызов метода WHAT на переменной \$lang и на имени класса Language на данный момент возвращает разные

результаты, в то время как в Pugs – одинаковые:

```
say $lang.WHAT;
# печатает 'Refs' в Rakudo
# и 'Language' в Pugs
say Language.WHAT;
# везде печатает 'Language'
```

Метод WHAT позволит узнать тип значения, которое сейчас хранится в переменной.

Например:

```
my $var = 'Perl 6';
say $var.WHAT;
$var = 6;
say $var.WHAT;
```

Первый вызов say \$var.WHAT напечатает Str, второй – Int.

Стоит обратить внимание на то, что если переменной присваивается результат вызова функции, то значение, возвращаемое WHAT, может оказаться не тем, которое мог бы ожидать программист:

```
$var = callme;
say $var.WHAT;
sub callme {
    say 'I am a sub';
}
```

Эта программа напечатает Int. Объяснение простое: значение именно этого типа возвращает оператор say:

```
say (say 2).WHAT;
```

Метод isa позволяет проверить тип переменной. Например:

```
say "OK" if 10.isa('Int');
my $var = 'string';
say "OK" if $var.isa('Str');
say "True" if (?100).isa('Bool');
say "True" if (Bool::False).isa('Bool');
```

Мультифункции

Мультифункции – набор функций с одинаковым именем, которые различаются типом и (или) количеством принимаемых аргументов. В объявлении должно присутствовать ключевое слово multi. Rakudo умеет различать такие функции:

```
multi sub say_time ($hour) {
    say "$hour:00";
}
multi sub say_time ($hour, $minute) {
    say "$hour:$minute";
}
say_time(14);
say_time(14, 15);
```

Try

В Rakudo реализован механизм перехвата исключений:

```
say 'before';
try {
    die 'Bye!';
}
say 'after';
```

Программа выводит обе строки 'before' и 'after', а сообщение об ошибке (в данном случае строка 'Bye!') содержится в переменной \$. Интересно отметить, что конструкция «проверить, не было ли ошибки» принимает такой вид: if !\$!.

Регулярные выражения

Rakudo поддерживает базовый набор для работы с регулярными выражениями. В частности, именованные регулярные выражения можно создавать с помощью ключевого слова regex. К таким объектам можно обратиться по имени, например, при вызове оператора сопоставления:

```
regex language {Perl|XML};
say "ok" if 'Perl' ~~ </language>;
say "not ok" unless 'PHP' ~~ </language>;
```

Остальные вопросы, связанные с реализацией компилятора Rakudo Perl 6, будут освещены в одном из докладов на конференции YAPC::Russia «May Perl 2008», которая состоится в Москве 17-18 мая.

Официальный сайт мероприятия – <http://2008.perlruussia.ru>.



Джонатан Вортингтон, один из активных разработчиков Rakudo, на украинском воркшопе «Perl Mova 2008» в Киеве. В частности, именно Джонатан реализовал поддержку ООП