Pragmatic Perl 7

pragmaticperl.com

Выпуск 7. Сентябрь 2013

Другие выпуски и форматы журнала всегда можно загрузить с http://pragmaticperl.com. С вопросами и предложениями пишите на editor@pragmaticperl.com.

Комментарии к каждой статье есть в html-версии. Подписаться на новые выпуски можно по ссылке pragmaticperl.com/subscribe.

Авторы статей: Владимир Леттиев, Станислав Пусеп, Андрей Шитов, Breno G. de Oliveira (garu)

Корректор: Андрей Шитов (ash)

Выпускающий редактор: Вячеслав Тихановский (vti)

Ревизия: 2015-01-17 12:58

Оглавление

1 От редактора

Конференция YAPC::Europe 2013 в Киеве завершилась! Читайте в этом номере отчеты с конференции. Видео постепенно добавляется на youtube, смотрите также интересное видео от Дмитрия Иванова. Фотоотчеты Ильи Чеснокова и Владимира Леттиева. Практически ко всем докладам добавлены слайды.

Мы продолжаем искать авторов для следующих номеров. Если у вас есть идеи или желание помочь, пожалуйста, свяжитесь с нами.

Приятного чтения.

■ *Вячеслав Тихановский*

2 Впечатления о YAPC::Europe 2013 в Киеве

Впечатления о конференции от участников

Анатолий Кочурков:

Я очень доволен тем, что имел честь побывать на таком большом мероприятии, как YAPC 2013. Ни на одной другой конференции, где я побывал, я не видел такого позитива и непринужденной, довольно азартной академичности, как здесь.

Каждый вечер, когда я выходил из здания, где проводилось мероприятие, я утверждался в понимании, что Perl жив и будет жить. Тем более, я это понимаю, когда вижу, что теперь Perl-сообществу есть что предложить в плане веб-фреймфорков.

Правда, я отстал от этой мысли года на два-три (а может и больше), потому что просто ленился узнать и увидеть, как много может предложить на сегодняшний день Perl.

Сегодня я влюблен в `Mojolicious`. И намерен основательно изучать его и пользоваться им для своих проектов. По крайней мере, его возможности меня крайне впечатлили. А особенно, его доступность. Да, конечно, как и в любом Perl-модуле понадобится определенная усидчивость для освоения его прелестей, но это окупится экономией времени и нервов — главное, увлечься.

Организация конференции была произведена на высоком уровне. Все было легко и доступно. Ребята-организаторы потрудились на славу и обеспечили не только конференцию, но и приятное времяпровождение.

Выводы после мероприятия следующие. Они касаются лично меня:

1) Не откладывать учить английский;
2) Не забывать мониторить свежие события Perl-сообщества;
3) Не стесняться задавать вопросы про Perl;
4) Общаться со всеми, кто в этом причастен.

Ларри смог суметь поселиться своим языком программирования в моей голове. А Андрей Шитов подсадил меня на этот язык. Не думаю, что смогу избавиться от этого наваждения.

Поэтому, как говорит и пишет Анатолий Шарифулин: use perl or die.

Дмитрий Иванов:

Начну с того, что YAPC::Europe 2013 была первой Perl-конференцей такого уровня, в которой я принимал участие. Уже через неделю после закрытия этой конференции я просматривал видео других ЯПЦов прошлых лет и отметил одну особенность: много людей говорят одно и то же: «Это моя первая конференция» (а Ларри Уолл добавил: «в Киеве»). А еще, эта конференция дала мне больше информации и больше новых знакомых, чем все предыдущие не-Перл-конференции вместе взятые…

Был минус *первый* день. И был Perl 6 хакатон. Хакатон — это когда все пишут код. А еще, как сказал Мориц, на хакатоне тебе подскажут и помогут. Лично я задолбал мистера Морица своими глупыми вопросами.

Еще бы. Когда смотришь в кусок кода, в котором символ «точка с запятой» переопределена как функция, чувствуешь, что начинает ехать крыша.

Отдельное спасибо Ивану aka Basiliscos — человеку, с которым мы писали тесты и который потратил первую половину Хакатона на то, чтобы я правильно задавал вопросы «команде Perl 6».

Затем был первый день… И второй, и третий. Промчались как одно мгновение. Писать подробно о докладах не буду — видео выкладывается, обзоры пишутся, рейтинги составляются. Очень хочу поблагодарить, сказать огромное спасибо организаторам. Ребята, вы постарались и, действительно, сделали конференцию международного уровня! Спасибо спонсорам! (надеюсь в этом журнале включат их

список ;)

И спасибо всем вам, перл-программисты!

Был на конференции Даниель — человек из солнечного Нью-Йорка. Он говорил по-украински лучше меня, хотя он в Украине был впервые. Был Мартон из Венгрии со слуховым аппаратом. А еще старые знакомые из Москвы и Баку — Алексей и Наим. Из Минска приехала целая каста Перл-программистов. Нидерланды были самые громкие.

Было незабываемо.

Здесь и сейчас.

I’m not hiring.

■

3 Ещё один отчёт о конференции YAPC Europe 2013 в Киеве

С 12 по 14 августа в Киеве прошла конференция YAPC::Europe 2013. Это ещё одна Perl-конференция, которая ежегодно проводится в различных городах Европы, и на этот раз переместилась в самую восточную точку — Киев (до этого самым восточным городом проведения конференции была Рига в 2011). Темой конференции стало «Будущее Perl» или «Perl будущего», ключевые доклады были посвящены вопросу о том, в каком направлении будет развиваться Perl, что сообщество ожидает увидеть в будущем Perl и, чёрт возьми, какая это будет версия: 5, 6 или вообще 7?

Перед конференцией

Первое мероприятие, которое было запланировано — хакатон про Perl 6, который прошёл за день до начала конференции в воскресенье. Хакатон должен был продлиться в течении 8 часов. На него я не попал, поскольку приехал позже, но люди, которые присутствовали, рассказывали, что провели отлично время, сидя бок о бок с Ларри Уоллом и решали различные задачи: фиксили несложные баги в баг-трекере Perl 6, писали для них тесты, дополняли документацию.

В тот же день в 19 часов была запланирована встреча всех участников конференции в одном из Киевских кафе — «Вареничная Катюша». Собственно именно социальные мероприятия стали одной из сильных сторон данной конференции, поскольку

позволяли участникам познакомиться, пообщаться и просто отлично провести время в дружной компании. Было непонятно, как более 100 человек уместятся в небольшом кафе, но они как-то умудрились это проделать. Основной потребляемый продукт — пиво, оказалось достаточно непросто получить — за ним выстраивалась очередь. Это, кстати, отлично дополняло общую атмосферу СССР 70-х, в которой была выполнена внутренняя обстановка кафе. В целом же вряд ли кто-нибудь испытывал неудобства, поскольку всё с лихвой компенсировалось тесным общением с Perl-хакерами со всего мира.

Первый день конференции

Конференция проходила в огромном здании международного конгресс-центра «Украинский дом» на Крещатике. В этом году в качестве билета на конференцию выступал распечатанный бэйджик, на котором помимо имени участника находился хитрый штрих-код, но на самом деле никому и в голову не пришло его проверять, поскольку посторонних людей сюда прийти просто не могло. Также каждый участник мог выбрать свой дизайн футболки: начиная от провокационного «use perl or die;», заканчивая нейтральным «25 years of Perl». Доклады проходили в трёх залах: большой основной зал и ещё два небольших зала для параллельных докладов. Кроме того, в холле перед главным залом, располагались удобные пуфики-мешки, в которых можно было плюхнуться и от-

дохнуть в перерывах между докладами. В холле также располагались стенды спонсоров конференции: *Booking.com*, *Reg.ru*, *Afnic*, *Provectus IT*, на которых активно работали хедхантеры, рассказывая о своих компаниях, раздавая различную атрибутику своих компаний собравшимся. Вообще, фраза хедхантеров *«we are hiring»* к концу конференции стала звучать так заезжено, что породила антимем *«we are NOT hiring»* в блиц-докладах, чтобы подчеркнуть, что речь пойдёт не о рекламе работодателя.

В целом обстановка была очень комфортной: работали кондиционеры, в холле всегда была вода. Во всех залах были доступны розетки, был доступен Wi-Fi (а со второго дня он стал даже более или менее сносно работать). Организаторы позаботились о питании участников: всех очень вкусно и быстро кормили обедами

и готовили настоящий кофе во время кофе-брейков.

Первый день докладов на конференции начинался с вступительного слова одного из главных организаторов конференции Андрея Шитова. По традиции, сразу было объявлено о месте проведения следующей YAPC::Europe в 2014 году. Один из участников комитета, Антон Березин, объявил о том, что среди трёх городов-кандидатов: Клуж-Напóка (Румыния), Гранáда (Испания) и София (Болгария) была выбрана София. Таким образом, счастливые обладатели шенгенской визы смогут в следующем году посетить YAPC::Europe 2014 в Софии.

Следующим выступлением был доклад Ларри Уолла «Вещи, о которых я узнал», который приехал в Киев значительно рань-

ше других участников и успел пройтись с экскурсией по городу. Поэтому первая часть доклада «Вещи, о которых я узнал за эту неделю» была посвящена Киеву и сопровождалась колоритными снимками: выбоин на дорогах, перегороженных строительным мусором пешеходных дорожек и прочих нелицеприятных картинок *опасного* Киева. Это было забавно. Вторая часть доклада была посвящена будущему Perl. Запомнилась цитата, если мне не изменяет память: *«Все языки программирования отстойные. Поэтому язык лучше, просто если он менее отстойный чем другой. В таком случае Perl реально крут, если сравнивать с Коболом. Даже PHP крут, потому что он круче чем, э..., чем HTML».*

Как известно, у Ларри был обнаружен рак. Часть доклада была посвящена этому факту. Ларри очень сильный и смелый человек,

он не боится говорить об этом спокойно и с юмором и показывает людям, что с этим можно бороться и не сдаваться. Резюмируя, он сказал, что за этот год шанс выжить значительно повысился.

Другие доклады первого дня

Дэйв Кросс (davorg) посвятил свой доклад 25-летней истории Perl, рассказав о тех вехах, которые были достигнуты за каждый из этих лет. Вероятно, 20 минут оказалось очень мало, поэтому многие страницы слайдов в конце были быстро пролистаны.

Аарон Крэйн (arc) рассказывал о NoSQL-базе данных `Redis`. Доклад был больше о самом `Redis`, т.е. без глубокой привязки к Perl. Было рассказано о том, как применяют `Redis` и в каких случаях он может быть

удобен.

Очень интересен был доклад Питера Рабитсона (ribasushi) о том, что бенчмарки — это реально сложное занятие. В ходе доклада он часто ссылался на статью «Adventures in Benchmarking» Дэвида Голдена, в которой был проведён эксперимент, когда проводился бенчмарк одной и той же функции, под разным именем (от *A* до *L*). Оказалось, что функция с именем *L* оказалась на 30% медленнее функции с именем *D*, таким образом, делался вывод: функции с именем *L* — это просто отстой и не стоит никогда их так называть. Шутки шутками, но это отлично показывает, что подходить к бенчмаркам нужно значительно серьёзнее. Существует масса обстоятельств влияющих на тест: это и поведение процессора, и фоновая работа операционной системы и других процессов в системе.

Кристиан Карг (odrm) рассказал, что Perl 6 мёртв, да здравствует Perl 5. Основной посыл доклада, о том, что именно Perl 5 — это рабочая лошадка, которая используется всеми в работе и именно он поэтому требует внимания и вложений со стороны сообщества и бизнеса.

Джон Йенсен (jon_jensen) сделал доклад «Adventures in Perl packaging», в котором рассказал об сборке Perl-модулей в виде rpm-пакетов для CentOS/Fedora. Он очень точно определил преимущества такого подхода к установке модулей перед *самосбором* на локальной машине с помощью `cpanm`: зафиксированные зависимости, включая зависимости на C/C++-библиотеки, что упрощает установку модулей; возможность создания репозиториев. Очень спорный момент доклада — это упаковка разных версий

Perl в `/usr/local` с последующей необходимостью хаков шебанга в исполняемых скриптах для переключения версии. По моему мнению это абсолютный нонсенс, но вероятно это решает проблему в каких-то нетипичных конфигурациях.

Сальвадор Фандино (salva) рассказал о создании удобных в сопровождении событийных программ с помощью модуля `Class::StateMachine`. Модуль основывается на понятии конечного автомата, когда система описывается в виде конечного набора состояний, переходы между которыми и программируются.

В своём докладе Пётр Коц рассказал о программе `snaked`, которая применялась в Яндексе в качестве альтернативы `crond` для тысячи сервисов.

Блиц-доклады первого дня

Апогеем первого дня стали блиц-доклады, когда десяток докладчиков в течении 5 минут презентуют свой модуль, проект или просто рассказывают интересные жизненные истории. Как мне показалась — это самая интересная часть конференции, поскольку тут не дают уснуть, периодически стуча по гонгу, да и сами докладчики очень живо и с юмором рассказывают доклады и, что важно, активно общаются с залом (ну и люди в зале тоже не отстают). Мне запомнились следующие доклады:

- Модуль Rex (система развёртывания).
- Модули Lingua::Num2Word Lingua::Word2
  для перевода чисел в слова и обратно
  для болеее 20 языков.
- Питер Раббитсон рассказал о том, что

нового в DBIx::Class 0.08250

- Saywer X в своей непревзойдённой манере, *ругаясь как пятилетний ребёнок* показывал слайды с котиками в докладе «Unstupidifying mp3 tagging»
- Алексей Капранов сделал очень спорный доклад о том, что молодой хипстер-питонщик круче, чем олдскульный Perl-зубр. На самом деле он говорил про то, что надо постоянно развиваться, уделять внимание тому, как твои программы интегрируются в рабочей системе и что оправдание *но это же работает на моей машине* не катит (*ок, тогда мы ставим твой рабочий ноутбук в продакшен*). Но кто ж это заметил?
- Дейв Кросс поведал нам о царстве слепых «The Kingdom of the Blind» — армии программистов, которые пишут на Perl, но не считают себя

Perl-программистами, не читают новостей о Perl, не посещают Perl-конференции, не общаются на форумах Perl Mongers. Присутствующие на данной конференции люди — это те лучи света в тёмном царстве, которые должны просвещать, рассказывать о современных практиках, обучать людей. Из доклада мне запомнился один приведённый пример, когда человек считал, что `eq` — это оператор присвоения строкового значения.

- Доклад Ларса Дийкова «Doors of Durin» я, честно говоря, не понял, что-то про Средеземье, эльфов и гномов. Но зато в конце было хорошо показано, что в Perl отличная поддержка UTF-8 и можно использовать даже языки средиземья ;-)

Первый день конференции завершил небольшой фуршет, спонсируемый Booking.com. И хотя планировалось, что он продлится полтора часа, люди не расходились три часа.

Важно отметить просто героическую работу организаторов и волонтёров конференции. Вячеслав Тихановский (vti) и Breno (garu_rj) весь день провели в холле, регистрируя участников, распечатывая бэйджики и раздавая футболки нужного цвета, размера и рисунка. Андрея Шитова можно было увидеть за тем, что он носил еду для ребят, так как у них не было времени, даже чтобы сходить поесть. Волонтёры обеспечивали работу всей техники конференции, начиная от микрофона, до Wi-Fi-точки, вели запись на видео всех докладов. Все были просто молодцы, за что им огромное спасибо.

Второй день конференции

Второй день конференции открыл Вячеслав Тихановский, рассказав о том, что сегодня в 19 часов нас ожидает экскурсия на самом большом теплоходе в Украине «Роза Виктория». Вместо 10 минут он говорил, кажется, всего минуты две, поэтому у следующего докладчика, оказалось гораздо больше времени на доклад.

Доклад Эндрю Мейна (zefram) «API design by counterexample» должен был рассказывать о том, как надо проектировать API объектного модуля, но превратился в жёсткую критику дизайна и документации модуля `DateTime`.

Следующие два доклада в зале №1 были посвящены Dancer, и я посчитал, что их очень важно посетить. Но, к сожалению,

доклады оказались просто неинтересными. Скучающий Sawyer X успел прямо во время их проведения сделать пару пулреквестов на Dancer и, разумеется, сообщил об этом в твиттере.

В том же зале далее был доклад Герберта Брюнинга (lichtkind) «Is your software development process complete?». Герберт изучает русский язык и по его мнению на русском языке доклад должен называться «Полное программирование». Речь шла о процессе работы над проектом. Предполагается, что цикл работы проходит через документирование, прототипирование, программирование, и в конце идут тесты. На вопрос из зала, почему тесты идут в конце, было отвечено, что тесты без кода будут бессмысленны, тесты проверяют какие-то граничные условия, и их трудно создавать, пока кода ещё нет.

Очень интересным был доклад Лиона Тиммермана (leont) «Replacing Module::Build». Речь пошла об истории развития систем сборки в Perl. Рассматривались `ExtUtils::MakeMaker`, `Module::Install`, `Module::Build` и `DistZilla`. Лион рассказал об их преимуществах, но больше о недостатках. Как известно в релизе Perl 5.20 уже не будет `Module::Build` — модуль, который в своё время пропагандировался как замена EUMM, сам оказался устаревшим. Как альтернатива предлагается переход на `Module::Build::Tiny`.

Сильное впечатление произвёл доклад Ричарда Джелинека (the whip) «HowTo: Perl as the most popular scripting language», основной девиз которого — *Use Perl! DAMMIT!*. Ричард представляет организацию *Propaganda.pm*, цель которой, как ясно из названия, — пропаганда исполь-

зования Perl. Грустно наблюдать, что Perl-сообщество, начиная с блогов хакеров, заканчивая целыми организациями, такими как Perl Foundation, используют в своих проектах `wordpress` (написанный на PHP), `mailman` (написанный на Python). При том, что существуют вполне жизнеспособные альтернативы, написанные на Perl. Вашу мать (вопрошает докладчик), вы собираетесь использовать Perl или нет?! Интересно, что даже организация Enlightenedperl использует `Drupal` на своём сайте, при этом каждый год зарубается финансирование проекта CMS на Perl, так как их и так много, зачем ещё. В общем, всё очень грустно, и если мы не начнём использовать Perl во всех проектах, где он отлично для этого подходит, мы никогда не сможем привлечь на свою сторону новых сторонников.

Мэт С Траут (mst) провёл зажигательный

доклад «State of the Velociraptor». Мэт, как настоящий шотландец, в национальном килте, с хриплым громогласным голосом, напоминает Мэла Гибсона в «Храбром Сердце». *Perl5 процветает, мы тащим туда самые клёвые фичи из Perl 6, Ruby и других языков программирования. Будущее прекрасно.*

Блиц-доклады второго дня

Как обычно, день завершили блистательные блиц-доклады.

- На эту конференцию по программе «Пришли новичка» (когда новичку оплачивают проезд на конференцию) попало двое человек: Михай Поп (Румыния) и Тео ван Хосл (Нидер-

ланды). По условиям программы они должны были сделать блиц-доклады, в которых и рассказали о своём пути в Perl и о том, как они попали на конференцию.

- Люкас Маи сделал доклад о своём модуле `Function::Parameters`. Модуль представляет собой реализацию сигнатуры функции для Perl, позволяя описывать параметры функций, их значения по умолчанию.
- Станислав Пусеп презентовал модуль `LWP::Protocol::Net::Curl`, который позволяет прозрачно использовать вместо `LWP` бэкенд `Net::Curl`, который поддерживает огромное число протоколов, работает с прокси и кэширует днс-запросы. В целом, насколько я понял докладчика, можно ускорить работу старых краулеров на `LWP` в два раза, не меняя ни строчки

кода.

- Наим Шафиев сделал доклад по терминалу PAC — графическому терминалу, для управления устройствами по ssh. К сожалению доклад был очень короткий и толком я не понял, какие у него возможности.
- Андрей Шитов рассказал о своём проекте YAPC.travel — веб-сайте, который позволит Perl-разработчикам, участвующим в различных конференциях, удобно бронировать отели или хостелы по самой низкой возможной цене.
- Микола Маржан проанонсировал несколько Open Source конференций, которые пройдут в Киеве и Украине в этом году.
- Морис Ленц рассказал о своём проекте irclog.perlgeek.de, который позволяет вести архивы irc-конференций.

- Tadeusz Sośnierz (tadzik) один из трёх участников локальной группы Варшава.pm рассказал, о том, как организовывал Polish Perl Workshop. Каждый шаг был достаточно прост, и нет ничего сложного в организации мероприятия — главное захотеть встретиться.
- Доклад Мэтью Уилсона (diakopter) назывался «Garbage Collection and Allocation in MoarVM», но в своём выступлении он делал очень странные заявление, вроде того, что через 2 года Perl 6 (Rakudo или MoarVM) будут эффективнее Perl 5. Шок! (хотя, возможно я что-то не так понял).

Круиз на теплоходе

Завершил второй день конференции Андрей Шитов с информацией о круизе по Днепру. Сначала он сказал, что потребуется быстро двигаться пешком 3 км до стоянки теплохода за полчаса. Но, к счастью это оказалось шуткой и всех организовано транспортировали на автобусах к пристани. Больше часа пришлось ждать отправления корабля из-за того, что некоторые участники застряли в пробке (да, оказалось в Киеве они тоже есть).

Круиз был просто фантастический. Все три палубы корабля были в распоряжении участников конференции и их партнёров. Бесплатные напитки, закуски и великолепный вид вечернего Киева с реки Днепр. Шутили, что если с кораблём что-нибудь случится, то это окажется непоправимым

ударом по Perl: на корабле Ларри Уолл и ещё почти 300 Perl-хакеров.

Третий день конференции

Третий день конференции ознаменовался почти пустым залом на первом докладе и небольшой очередью к кулеру с водой. Очевидно, что круиз удался.

Самым интересным докладом третьего дня стало обсуждение будущих версий Perl. Андрей Шитов сразу дал понять, что упоминание *Perl 7* — это своего рода привлекающий внимание маячок, который может провоцировать плодотворное обсуждение о будущем Perl, а не предложение кардинально изменить номер версии Perl. Андрей рассказал историю появления

последней дискуссии по Perl 7 с блог-поста Ovid'a, который спровоцировал флейм о том, надо или не надо переименовывать Perl 5. С одной стороны, Perl 5 появился почти 20 лет назад, более 10 лет назад появился Perl 6, который задумывался как развитие Perl 5. Но реальность оказалась такова, что Perl 6 так и не стал готовым для реального использования, а Perl 5 оказался в версионном тупике — он продолжает развиваться, но только в рамках версии 5. Для любого постороннего наблюдателя 5 всегда меньше 6 и видя состояние Perl 6, создаётся впечатление, что Perl в целом просто стагнирует. Нужен прорыв, отсюда и разговор, что надо дать новое название Perl 5, чтобы дать ему возможность развития, так, чтобы это было заметно для окружающего мира. Но с другой стороны, Perl 7 или другое имя подразумевает какое-то сильное отличие, серьёзное из-

менение API, чего не будет, если просто переименовать Perl 5. Все скажут, — тут ничего нового, это всё тот же «древний» Perl 5. Кроме того, такое переименование поставит крест на разработке Perl 6 по той же причине: 6 это меньше, чем 7. Таким образом, такое переименование пойдёт во вред обоим проектам.

Также во время доклада было продемонстрировано видео-интервью с брайеном ди фоем о будущем Perl, перевод которого есть в шестом номере журнала Pragmatic Perl. В конце доклада Андрей пригласил на сцену Ларри Уолла, Лиз Маттейсен и Карла Мэсака. В основном речь пошла о перспективах Perl 6. Ларри Уолл сказал, что Perl 5 — это один язык, Perl 6 — другой, он приветствует создание форков, как Perl 5, так и Perl 6. Пусть будет больше экспериментов — создавайте свой Perl и давайте

ему любую версию, какую захотите. Всё это будет одна большая группа Perl языков и не надо среди них выделять один самый **главный** Perl.

В итоге, в вопросе будущего Perl сохраняется *Status Quo*: потихоньку ковырять Perl 5, временами играться в Perl 6 и наблюдать, как Perl потихоньку разделяет судьбу Кобола. Ок.

Следующий доклад был у Питера Раббитсона «SQL metaprogramming — non-ORM uses of DBIx::Class», в котором рассказывалось об интересных возможностях `DBIx::Class` для генерации сложных SQL-запросов.

Также я посетил интересный доклад Zefram'а о том, почему временные зоны это сложно. Очень обстоятельно было рас-

сказано об истории появления временных зон. О том как создание быстрых средств передвижения (поезда) создало серьёзную путаницу с локальным временем и времени на железнодорожных станциях. О том, как это преодолевалось. Как создавалось всемирное координированное время, и как тяжело страны принимали его. Интересно, что временных зон значительно больше чем 24 (кажется 35), и отличие от гринвичского времени может быть не кратно 1 часу (например +5:30 для Индии).

После ланча был замечательный доклад Sawyer X «Asynchronously Fantastic». Сойер рассказал о принципах асинхронного программирования, об `AnyEvent`. В качестве примера он рассказал о программе, которая делает запрос на сайт IMDB об актёрах, которые играют в различных популярных сериалах. Таким образом он хотел

получить список актёров, которые играли сразу в нескольких сериалах. Посчитав, что если каждый запрос будет выполняться в среднем 2 секунды, то последовательный запрос всех страниц по всем сериям всех сезонов всех сериалов займёт несколько дней. В то время как его программа на `AnyEvent` справилась с задачей за несколько минут.

Следующий доклад, который я посетил, был у Рейни Урбана «Recent compiler optimizations», в котором он делился своими успехами в оптимизирующем компиляторе B::CC для компиляции Perl кода в C, а затем в исполняемый код. Рассмотрены различные виды оптимизации — это и разворачивание циклов, у которых известно количество итераций. Отключения проверки на существование элемента ключа с автоматическим его созданием при обращении к вложенному ключу (no-autovivify).

Отключение проверки на наличие магии (no-magic). На тестовом примере, где решение на чистом Perl'е выполнялось за 23 минуты, все эти оптимизации давали повышение скорости выполнения до 2,5 минут (почти на порядок).

Был ещё один доклад на тему замены crond и тоже в Яндексе — «Reliable cron jobs in distributed environment» Олега Комарова. Речь в докладе идёт об утилите `switchman`, которая позволяет запускать различные задачи на кластере из нод Apache Zookeeper, контролируя выполнение на определённых нодах через распределённую систему блокировок (запуск только на одном сервере или запуск на сервере, на котором доступен процессор или нужный объём памяти для задачи).

Блиц-доклады третьего дня

После кофе-брейка настало время блиц-докладов, и на этот раз выступало рекордное число участников — 13. Вот самые интересные:

- Доклад о `PkgSrc` — системе сборки для 11 платформ (и прежде всего NetBSD) Дженса Резака. Собственно, опять полезный совет — пакуйте ваши Perl-модули в той системе сборки, которая работает на вашей операционной системе.
- Sawyer X вышел буквально на минуту, чтобы продемонстрировать новый дизайн для сайта dbix-class.org. Новый дизайн безусловно просто фантастический, вероятно в скором времени мы увидим его вживую

(старый сайт пока на месте).

- Breno Oliveira рассказал о том, что теперь можно слать отчёты о тестировании модулей CPAN, даже пользуясь современным инсталлятором `cpanm`.
- Дмитрий Иванов продемонстрировал в работе свою CMS — Simpleness, написанную конечно же на Perl.
- Брайн МакКолли похвастался, что выиграл турнир по Perl Golf, который устраивал Reg.ru. Его решение для задачи игры в Go содержало всего 205 символов.
- Очень приятно было слушать доклад Руслана Закирова о создании новой фичи для Perl 5: копирование пар ключей/значений из одного хэша в другой с использованием краткой записи my %r = %h{qw(a b c)}. Это потребовало изменения в синтаксисе Perl и изменений в самых недрах Perl

5. Но, как оказалось, разобраться с этим может быть вполне по силам любому Perl-хакеру, надо только не бояться этого. Новый синтаксис станет доступен, начиная с Perl 5.20.

- Илья Чесноков честно рассказал о своём пути к программированию на Perl. Если ты живёшь не в Москве, а в Рязани, то работы для Perl-программиста найти практически нереально. Илья поблагодарил Reg.ru и вообще все компании, которые дают шанс таким ребятам как он работать с Perl удалённо.
- Иван Фомичев провел Perl Quiz — викторину на знание Perl. Безусловно, каждый вопрос был с подвохом, и многие (и конечно я сам) узнали для себя много нового. Был спорный вывод об использовании `Readonly` вместо модуля `constant` — доста-

точно взглянуть в баг-трекер, чтобы понять, что он тоже не сахар.

- Стефан Сейферт (Nine) с докладом «use perl or die;» рассказал о двух своих проектах: `CiderCMS` — Perl CMS и `CiderWebmail` — реализация веб-интерфейса для почты на фреймворке Catalyst. Это хороший ответ на доклад участника Propaganda.pm — всё-таки на Perl есть достойные альтернативы популярным веб-приложениям.
- Пирс Колей (pdcawley) поразил всех удивительным исполнением песни о конференции и присутствующих известных людях. Оказывается, у Perl-хакеров есть и другие таланты.

Закрытие

Андрей Шитов в заключительном слове поблагодарил всех за участие в конференции, представил всех тех ребят, которые работали над организацией и проведением мероприятия. Это была фантастическая конференция. Спасибо всем участникам и организаторам за доставленное удовольствие! Ждём видеозаписей докладов, чтобы дать возможность тем, кто не присутствовал, увидеть всё своими глазами.

■ *Владимир Леттиев*

4 YAPC::Europe 2013 — уникальная возможность

Израильтянин, болгарин, немец, украинец и бразилец заходят в бар...

Когда мы говорим о Perl-мероприятиях, то YAPC::Europe это одно из тех, которое надо обязательно посетить. И это не только по очевидным причинам, что конференция самая большая в Европе, по отличным докладам, демонстрациям последних и крутых приложений, модулей и языка вообще. Кроме всего этого, на YAPC::Europe вы лично переживете опыт нахождения со многими людьми из разных стран в одном месте, разделяя с ними общую цель и позицию. Увлечение программированием. Увлечение перлом.

Это может прозвучать обычным для европейца, но подумайте как это переживается теми, кому это вдиковинку в разных частях света. Возможность видеть как общаются люди из разных культур, как делятся знаниями, смеются — и, признаемся, выпивают много пива — действительно невероятно. С более чем 300-ми участниками из 37 разных стран YAPC::Europe это как минимум очень богатый опыт.

Другой замечательной и уникальной особенностью YAPC::Europe является то, что конференция перемещается каждый год. Это значит, что вы не только встречаете множество интересных людей, но и посещается новую страну каждый раз. В этом году, в 14-й раз, международное Perl-сообщество посетило Украину, где в красивом городе Киеве и проходила конференция.

Украинский дом был выбран в качестве места проведения для YAPC, с его великолепным — и огромным — залом на первом этаже. Сразу возле зала организаторы позаботились о нескольких разноцветных пуфиках, создав импровизированную лаунж-зону, что было весьма популярным среди хакеров во время всей конференции. Два других зала, чтобы вместить все три потока, располагались на втором этаже, ланчи и кофе-брейки — на пятом. Другим хорошим аспектом здания было его расположение, можно было дойти пешком до центральной площади, ресторанов, пабов и ночной жизни.

Как обычно на Perl-мероприятиях, веселье началось еще до самой конференции. В воскресенье Jonathan Worthington проводил хакатон, посвященный Perl 6, в стилизованном под средние века конференц-зале

отеля «Днипро». После этого все переместились в вареничную «Катюша», где подавали очень вкусную местную еду.

Первый день начался слегка скомканно, были некоторые проблемы с печатанием билетов, но участники быстро нашли вход в главный зал, где Андрей Шитов (ash) открыл конференцию и пригласил на сцену Ларри Уолла, чье выступление в очередной раз было настолько проницательным, насколько оно может быть от автора языка Perl. После этого как всегда интересный Dave Cross дал отличную презентацию, рассказав о двадцапятитилетней истории Perl за двадцать пять минут. Затем конференция разделилась на три потока с докладами от бенчмарков до двух-факторной авторизации, и каждый хакер был предоставлен сам себе. В конце дня R Geoffrey Avery провел первую серию блиц-докладов, когда

люди поднимаются на сцену с пятиминутным докладом и говорят в общем-то о чем угодно. Затем все переместились в центральный зал на вечеринку, которую спонсировали Booking.com. Это была прекрасная возможность встретиться со старыми друзьями, познакомиться с новыми людьми и вообще классно провести время.

Во второй день конференции были доклады о дизайне API, ZeroMQ, Dancer, DevOps, Log4perl, PSGI, сериализации, обработке ошибок и о многом, многом другом. Вечером выступил Matt S. Trout со своим докладом о состоянии Велосираптора, затем была еще одна сессия блиц-докладов.

После этого для всеобщего удовольствия для участников, организаторы привезли всех на огромный теплоход для речного

круиза, где каждый мог наслаждаться прохладным ветерком, великолепными видами, едой и напитками, наблюдая закат и общаясь с хакерами. Этот вечер определенно всем запомнился.

Последний день YAPC также был наполнен интересными докладами о Perl и CPAN, с темами о параллельных вычислениях, асинхронном программировании, регулярных выражениях и оптимизациях компилятора, летающих роботах (!), тестировании и о многом другом. Были также нетехнические доклады о таких важных темах как рост Perl-команд и как общаться с менеджерами, и, конечно, последняя сессия блиц-докладов. После этого Андрей Шитов закрыл конференцию, благодаря всех участников, спикеров, спонсоров и волонтеров за еще одну клевую конференцию.

В холле во время конференции спонсоры раздавали разные подарки, рассказывали про свои удобные для хакеров рабочие места, пытаясь привлечь талантливых Perl-программистов к себе на работу. На другой стороне холла Wendy “woolfy” van Dijk раздавала всем деревянные туиты (tuits) — известная игра слов от “getting around to it” до “getting a round tuit”. Поэтому если кто-то хотел над чем-то похачить, но не хватало времени, просто можете передать ему этот круглый туит, и у него уже не будет никаких отговорок!

Среди других плюсов можно отметить отсутствие критических задержек в расписании — что очень удивительно, учитывая 60 докладов в трех разных залах — и футболки с выбранными участниками цветами и надписями. К сожалению, по разным причинам не всем они достались

(но дизайны всех футболок выложены в открытый доступ — *прим. перев.*). Интернет периодически глючил, но организаторы постоянно общались с провайдером и пытались улучшить ситуацию. К счастью, участники относились с пониманием, и это не стало помехой.

Если вы никогда не были на YAPC::Europe, вам обязательно стоит попробовать посетить ее. Как было объявлено в Киеве, следующим городом станет София, столица и самый большой город Болгарии. Организаторы уже начали работать над конференцией, чтобы сделать ее такой же крутой, как и эту. Увидимся там!

■ *Breno G. de Oliveira*

5 Пост-пост про YAPC::Europe 2013 в Киеве

В августе этого года в центре Киева состоялась ежегодная международная конференция YAPC::Europe 2013. В этом номере журнала опубликовано несколько отчетов, написанных участниками, а в заметке Владимира Леттиева прокомментированы доклады, которые он успел послушать. Я не был почти ни на одном, поэтому расскажу про то, как выглядит конференция глазами организаторов.

Мы начали готовить конференцию в июле прошлого года. Началось все с невинного вопроса о том, не провести ли нам эту конференцию в Киеве.

Вопрос:

From: Andrew Shitov

Date: 2012/7/10

Subject: йапси?

To: Viacheslav Tykhanovskyi, Yaroslav Korshak

Привет!

Что думаете про YAPC::EU в Киеве?

Ответ:

From: vti

Date: 2012/7/10

Subject: Re: йапси?

To: Andrew Shitov

Дык я за! :)

YAPC::Europe перемещается между европейскими городами. Существует так называемый Venue Committee, который выбирает, где будет следующая конференция. Комитет состоит из организаторов прошлых мероприятий, а выбирает он из заявок, которые подают желающие из разных городов. На 2013 год в срок не было подано ни одной заявки. Я не собирался делать конференцию в Киеве, потому что пришлось бы делать ее лучше, чем конференцию в Риге 2011 года. Но поскольку ни одна европейская страна не подала ни одной заявки, миссия изменилась. Мы не только получили право проведения конференции, но еще заодно спасли и YAPC::Europe, и Perl. (Надо ли говорить, что после нас на 2014 год было подано сразу три заявки, две из которых частично повторяют текст и структуру нашей.)

На YAPC::Europe обычно приезжает около трехсот человек. Мы нашли в Киеве семь помещений, где можно было бы их разместить и организовать несколько потоков докладов. Некоторые варианты были однозначно со скамьи запасных, но два-три были очень хорошими, причем цена отличалась в два-три раза. Мы выбрали лучшее по размеру и по месторасположению помещение в самом центре Киева, поэтому весь следующий год пришлось работать со спонсорами :-) Но без Украинского Дома конференция была бы менее яркой.

Сама по себе конференция длится три дня, однако последние несколько лет программа выходит за эти рамки и может длиться до недели. В нашем случае был один дополнительный день, воскресенье, в который прошло два мероприятия: хакатон про Perl 6 и предварительная встреча

участников в кафе.

Предварительная встреча

Вечером в воскресенье была предварительная встреча участников в вареничной «Катюша» на Владимирской улице недалеко от Крещатика и от места проведения конференции. Когда я нашел место и сделал предварительный заказ на сто человек, мы сразу же написали новость и попросили всех желающих зарегистрироваться на сайте. В течение первых двух-трех недель регистрировались очень медленно: казалось, что заказ окажется слишком большим. Зато за неделю до конференции все начали регистрироваться с такой скоростью, что пришлось увеличить заказ вдвое, а зарегистрированных посетителей стало

155 (а регистрировались не все). Мы заняли не только весь второй этаж «Катюши», но и веранду соседнего кафе.

Постоянные посетители YAPC::Europe говорят, что раньше такого не было. Вечером накануне конференции можно было встретить в городе человек пять-десять знакомых. А в последние годы все больше и больше участников стараются приехать заранее, и предварительные встречи становятся все масштабнее.

Хакатон про Perl 6

На хакатон собралось около 25 человек, среди которых были все основные разработчики (кроме Патрика Мишо и Дамиана Конвея, которые в этом году не смогли

приехать) и просто интересующиеся, которым хотелось посмотреть на процесс разработки изнутри. Все прошло в одном из конференц-залов отеля «Днипро». Дверь в зал вела из ресторана, поэтому на кофе-брейки далеко ходить было не нужно.

— Мой первый вопрос Карлу Мэсаку. Что вы там делали, чего достигли, завершили ли вы Perl 6?

— Да, мы завершили, но просто никому не сказали. Всегда приятно видеть и старые лица, и новые. Да и зал был хороший. В течение дня я видел, как участники работали над багами (на IRC-канале говорили: «О, вот этот баг можно закрыть»), над реализацией или писали первые строки на Perl 6. Или просто сидели и разговаривали про синтаксис и семантику. Этим и был занят весь день. Столько людей, общающихся по поводу Perl

6; довольно приятно. Так что делали всё: от собственно работы над компилятором до создания модулей. Кажется, я видел двух-трех человек, которые написали модуль и загрузили его куда-то. Так что много чего произошло.

Несмотря на все перипетии с Perl 6, я продолжаю собирать хакатоны. Лично мне интересно, чтобы люди работали над документацией и выявляли нестыковки и неаккуратные места. По поводу компилятора, скорости его работы и скорости разработки как-нибудь в другой раз :-)

Дискуссия про Perl 6

Конференция называлась Future Perl («Будущий Perl»). В среду я провел 40-минутное

обсуждение про это самое будущее. Я рассказал о флейме про Perl 7, который возник в начале этого года, о том, что можно было бы с этим сделать, показал видеозапись с брайаном ди фоем, а потом пригласил на сцену Ларри Уолла, Лиз Маттейсен и Карла Мэсака. Полностью обсуждение доступно на Ютубе. По поводу смены версии Ларри высказал разумную идею о том, что современная тенденция в языках такова, что смена версии означает потерю совместимости с предыдущей. Из положительного — разработка Perl 6, возможно, опять будет ускорена, хотя как все получится, я не берусь предсказывать. Одним словом, посмотрите и делайте выводы.

Футболки

Традиционно все участники получают футболку с надписью о том, где они были. В этом году мы реализовали ноу-хау и предложили всем участникам выбрать не только размер, но и цвет и надпись на футболке. Такого еще никто не делал. На лицевой стороне у всех написано «YAPC::Europe, Kiev, Ukraine, 12–14 August 2014», а на спине — один из вариантов на выбор:

- 25 years of Perl
- use perl or die;
- Pragmatic Perl
- Perl 5
- Perl 6
- Perl 7
- Perl?
- Perl!

- Modern Perl
- Rakudo Perl
- Camelia Perl
- Future Perl

Свой выбор сделали около 200 человек. В сочетании с размерами и цветами получилось более 120 уникальных комбинаций, так что этот заказ оказался непростым и для компании, которая его выполняла. Но зато напечатали быстро, как и обещали, и мы со Славой уже в пятницу завалили мой номер в гостинице коробками с футболками, пытаясь их немного отсортировать и отобрать футболки для участников хакатона.

```
perl -MData::Dumper -nE'chomp; $t
    {$_}++; END{say Dumper \%t}'
```

Самым популярным вариантом оказался слоган `use perl or die;` (выбрало 94 человека), за ним идет нейтральная надпись «25 years of Perl» (43). Все остальные надписи следуют с большим отрывом. Двенадцать человек выбрали «Pragmatic Perl».

Эксперимент с футболками получился очень интересным и удачным, хотя он весьма трудоемок, а стремительный рост числа участников за три-четыре дня до конференции не позволил нам напечатать достаточный запас. Но зато мы опубликовали векторные исходные файлы, и теперь любую футболку можно напечатать в любой момент.

Речной круиз

«Кульминацией второго дня YAPC::Europe стал речной круиз, киевская версия традиционного ужина. Это был довольно большой корабль, идущий по Днепру, с едой, напитками и множеством перловиков. Поскольку не было стационарных столов и из-за того, что приходилось ходить за едой и напитками, народ перемешивался и общался больше, чем на традиционных обедах. Мне это очень понравилось». (Мориц Ленц)

На одной из предыдущих YAPC::Russia в Киеве мы уже пробовали кататься на корабле, и было заранее известно, что поездка будет приятной. Для YAPC::Europe мы арендовали трехпалубный прогулочный теплоход, по заверениям владельцев он самый большой не только в Киеве, но и во всей Украине. Как и в истории с предвари-

тельной встречей, пришлось увеличивать число заказанных порций еды, причем потребовалось сделать это трижды. На корабле было около 350 человек: 330 участников конференции и их семьи (многие были даже с маленькими детьми).

Если вы не были на конференции и на круизе, посмотрите видео Дмитрия Иванова — значительная часть этого видеоролика (начиная с 5:34) посвящена круизу.

Афтерпати

В этот раз каждый вечер был занят какой-нибудь развлекательной программой. В минус первый день предварительная встреча, во второй — круиз, а в первый и третий дни спонсоры устраивали для всех участников

небольшой ужин-пати. Сначала орешковая вечеринка прямо в помещении, а потом зависон в «Пивной бочке».

Мне кажется, это должно стать традицией на следующих YAPC::Europe. Раньше был занят только один вечер, когда проходит традиционный ужин, а теперь можно предлагать спонсорам занимать и другие свободные вечера. Все равно участники обычно тусят группами до позднего вечера. (Надеюсь, в будущем не потребуется делать посреди конференции выходной день, чтобы участники могли отоспаться после афтерпати.)

Другие придумки

Часть того, что было реализовано, было или придумано, или отработано на предыдущих мероприятиях. Что-то мы придумали впервые. Но что-то реализовать не удалось. Просто потому, что не успели. Например, в 2011 году в Риге стержнем конференции был упор на удобство для докладчиков. Для них был организован четырехчасовой тренинг накануне конференции и были подготовлены две отдельных комнаты (Speakers rooms), где можно было в тишине доделать свой доклад и даже подключить ноутбук к проектору и посмотреть, что вышло. В этом году целью было показать, как хорош Киев: поэтому была и партнерская программа, и речной круиз, и помещение в самом центре. За бортом осталась, например, групповая фотография со всеми участниками на ступеньках Украинского

Дома. Зато получился эксперимент с публикацией новостей о конференции раз в неделю в течение всего года подготовки.

Волонтеры

В этом году в проведении конференции нам помогали аж семеро волонтеров. Трое из них (Breno Oliveira из Бразилии, Amalia Pomian и Diana Donca из Румынии) работали за столом регистрации и раздавали именные беджи и футболки. Четверо (Евгений Патлан, Вячеслав Саржан, Александр Кубрак и Микола Маржан из Украины) — обеспечивали строгость расписания, помогали докладчикам с оборудованием и вели видеозапись. Без них было бы все намного сложнее. Кстати, видеозаписи уже появляются на канале youtube.com/yapctv.

YAPC::Europe 2014

В следующем году YAPC::Europe пройдет в Софии. Другие две заявки, полученные от испанского города Гранада и румынского Клуж-Напока, тоже были сильными и интересными предложениями, поэтому хочется надеяться, что в 2015 году они будут повторены.

Конференцию в Софии проводит Мариан Маринов, который в течение пяти лет собирает там болгарский Perl-воркшоп (ОК-ОК, первый в 2009-м году мы делали с ним вдвоем). Приглашаю всех побывать в Софии и не пропускать очередной шанс встретиться с Ларри Уоллом и многими другими любителями (но при этом профессионалами) перла. Заявленная тема конференции — Distributed Perl.

Отдельной строкой

Отдельной строкой я выражаю благодарность Вячиславу Тихановскому (vti), с которым мы готовили эту конференцию, бегали по Киеву по разным конференционным делам, встречали в аэропорту гостей, читали и писали в твиттер и на сайт, придумывали ход событий, ночью правили расписание и пару раз за все время успели пообедать.

■ *Андрей Шитов*

6 Статический анализ кода

Зачем?

С одной стороны, выразительность языка Perl позволяет писать код со скоростью мысли. С другой, запросто можно наставить пару-другую «подводных мин», которые проходят через все тесты (если они у вас есть) и тем не менее «взрываются», отображая *errors* и *warnings*, мало связанные с сутью проблемы.

А что, если заставить Perl заняться самоанализом?

Perl::Critic

Без сомнений, самый полный и самый важный инструмент из семьи статического анализа Perl. Утилита perlcritic исследует исходный код и оценивает его, следуя правилам из Perl Best Practices, помимо других, таких как, например, цикломатическая сложность.

Присутствуют элементы игрофикации: «уровни сложности» от 1 до 5 (называющиеся, соответственно: *brutal*, *cruel*, *harsh*, *stern* и *gentle*).

Например, вот результат запуска `perlcritic -3 test.pl`:

```
1 Code not contained in explicit
    package at line 1, column 1.
    Violates encapsulation.  (
    Severity: 4)
```

```
Found use of Class::ISA. This module is deprecated by the Perl 5 Porters at line 4, column 1.  Find an alternative module.  (Severity: 5)
Found use of Switch. This module is deprecated by the Perl 5 Porters at line 5, column 1.  Find an alternative module.  (Severity: 5)
Module does not end with "1;" at line 5, column 1.  Must end with a recognizable true value.  (Severity: 4)
```

Важно понять, что Perl::Critic никому ничего не навязывает, так же как и не пропагандирует некий стандартный диалект «Modern Perl»! Его всегда можно запросто «успокоить»:

```
my $const_value = eval $const_name . '()'; ## no
```

critic

Только на уровне *harsh* требуется доказать `Perl::Critic`, что понимаешь, что делаешь (при этом учитывается область видимости как у прагм):

```
my $const_value = eval {
    ## no critic (
       ProhibitNoStrict
       ProhibitNoWarnings)
    no strict qw(refs);
    no warnings qw(once);
    return *$const_name->();
};
```

Для личных настроек и исключений имеется файл `~/.perlcriticrc`:

```
severity = harsh

[TestingAndDebugging::
   RequireUseWarnings]
```

```
equivalent_modules = common::sense

[TestingAndDebugging::RequireUseStrict]
equivalent_modules = common::sense
```

Совет: полезно ознакомится со списком policies, которые присутствуют в дефолтовым дистрибутиве `Perl::Critic`. Помимо них, имеется Perl::Critic::Pulp, сборник других полезных *policies*, например, Perl::Critic::Policy::Modules::ProhibitModuleSheb (запрет на использование хэшбэнга `#!/usr/bin/perl` в файлах типа `.pm`). Для использования этой *policy* достаточно добавить в `~/.perlcriticrc`:

```
[Modules::ProhibitModuleShebang]
severity = gentle
```

При написании тестов для своих модулей можно воспользоваться Test::Perl::Critic (не забывая при этом проверять в таком тесте наличие `$ENV{AUTHOR_TESTING}`!).

Ну и, если лень устанавливать из CPAN, имеется в наличии *webservice* perlcritic.com, *à lá* validator.w3.org.

Perl::MinimumVersion

С кем не случалось: у тебя код работает, а у `$whoever` - нет? Иногда это всего лишь наличие `$a //= 1` или `map { s/^\s+|\s+$//gr } @b` в одной единственной строке здоровенной программы. Утилита perlver помогает двухсторонне: она показывает, на какую версию Perl рассчитывал разработчик, а также место потенциальных

«эксцессов» в коде. Например, вот результат запуска `perlver --blame test.pl`:

```
1 ---------------------------------------------------------------

2 File    : test.pl
3 Line    : 8
4 Char    : 4
5 Rule    : _perl_5010_operators
6 Version : 5.010
7 ---------------------------------------------------------------

8 //=
9 ---------------------------------------------------------------
```

Для своих тестов можно использовать Test::MinimumVersion (в этом случае более подходит ограничение на `$ENV{RELEASE_TESTING}`).

Кстати: иногда corelist служит интересным дополнением `perlver`. Например, чтобы узнать, с каких пор в Perl присутствует синтаксис `given/when`, запускаем `corelist --feature switch`:

```
Data for 2013-08-11
feature "switch" was first
    released with the perl v5.9.5
    feature bundle
```

Perl::PrereqScanner

Еще один грешок - не оповещать свои *dependencies* (или оповещать, но не полностью). Dist::Zilla делает это автоматом и притом чересчур эффективно: запросто притесывается всякая чепуха, без которой легко можно обойтись (например, заменив Text::Trim на `s/^\s+|\s+$//g`). К

счастью, можно сканировать и вручную, посредством scan_prereqs:

```
scan_prereqs --combine lib/ t/
```

```
Carp                = 0
Config              = 0
Fcntl               = 0
HTTP::Date          = 0
LWP::Protocol       = 0
LWP::UserAgent      = 0
Net::Curl::Easy     = 0
Net::Curl::Multi    = 0
Net::Curl::Share    = 0
Scalar::Util        = 0
base                = 0
strict              = 0
utf8                = 0
warnings            = 0
```

А Dist::Zilla можно научить «пропускать» только модули из стандартного (CORE) дистрибутива Perl при помощи `Dist::`

`Zilla::Plugin::OnlyCorePrereqs`.

Test::Mojibake

scan_mojibake из этого дистрибутива помогает избегать проблем с кракозябрами, санкционируя употребление `use utf8` / `=encoding utf8`:

```
not ok 13 – Mojibake test for t/bad/bad–latin1.pl_
#   Failed test 'Mojibake test for t/bad/bad–latin1.pl_'
#   at /Users/stas/perl5/perlbrew/perls/perl–5.16.2/lib/site_perl/5.16.2/Test/Mojibake.pm line 168.
# Non–UTF–8 unexpected in t/bad/bad–latin1.pl_, line 6 (source)
```

```
not ok 14 - Mojibake test for t/bad/bad-utf8.pl_
#   Failed test 'Mojibake test for t/bad/bad-utf8.pl_'
#   at /Users/stas/perl5/perlbrew/perls/perl-5.16.2/lib/site_perl/5.16.2/Test/Mojibake.pm line 168.
# UTF-8 unexpected in t/bad/bad-utf8.pl_, line 5 (source)
```

Test::Vars

Если GCC предупреждает о присутствии неиспользованных переменных, то что мешает Perl делать то же самое?! Только лишь отсутствие Test::Vars в стандартном дистрибутиве. Увы, у этого модуля нет утилиты *stand-alone*. Тем не менее, можно запускать его таким вот образом:

```
perl -MTest::Vars -e 'all_vars_ok()'
```

Test::Vars собирает имена анализируемых файлов из MANIFEST, поэтому если такого файла нет, фокус не удастся. Вот пример вывода:

```
# $result is used once in &LWP::Protocol::ftp::__ANON__[lib/LWP/Protocol/ftp.pm:307] at lib/LWP/Protocol/ftp.pm line 278
# $mtime is used once in &LWP::Protocol::ftp::request at lib/LWP/Protocol/ftp.pm line 357
# $mode is used once in &LWP::Protocol::ftp::request at lib/LWP/Protocol/ftp.pm line 357
not ok 14 - lib/LWP/Protocol/ftp.pm
#   Failed test 'lib/LWP/Protocol/ftp.pm'
```

```
#   at -e line 1.
```

Pod::Coverage

Неполная документация хуже отсутствующей документации. Поэтому Pod::Coverage сканирует определения функций и методов в исходном коде и проверяет наличие соответствующей документации POD. Утилита `pod_cover` также зависит от файла `MANIFEST`:

```
Summary:
 sub routines total     : 63
 sub routines covered   : 56
 sub routines uncovered: 7
 total coverage         : 88.88%
```

И, конечно же, имеется модуль Test::Pod::Cover

Все вместе

Созданный для настоящих Perl-программисто
`Dist::Zilla::PluginBundle::TestingMan`
собирает некоторые из этих модулей (и
многие другие).

■ *Станислав Пусеп*

7 Сборка deb-пакетов модулей Perl для Debian и Ubuntu

Несколько докладов прошедшего YAPC::Europe 2013 были посвящены теме сборки пакетов (rpm, pkgsrc) модулей Perl для упрощения процесса установки Perl-программ с использованием штатного менеджера пакетов системы. Чтобы дополнить спектр охваченных пакетных менеджеров, в данной статье будет рассказано о сборке deb-пакетов, которые применяются во множестве популярных Linux-дистрибутивов, таких как Debian и Ubuntu.

Зачем паковать Perl-модули в пакеты?

Большинство Perl-программистов знакомы с привычными утилитами для установки Perl модулей: `cpan` или `cpanm`. Эти утилиты очень удобны для разработчиков и позволяют им достаточно легко установить нужные модули в системный или домашний каталоги. К сожалению, подобный метод установки плохо пригоден для развёртывания уже готового проекта, поскольку набор устанавливаемых модулей мог обновиться на CPAN, и обновлённые модули могли оказаться неработоспособными или могли сломать работу зависимых модулей. Даже развёртывание собственного CPAN с использованием таких систем как `Pinto` для фиксации версий может быть непригодно, если администратор не хочет,

чтобы на сервер, на котором будет работать продукт, нужно было устанавливать компиляторы, заголовочные файлы всех сторонних библиотек, которые потребуются для сборки XS-модулей, и следить за тем, чтобы после обновления системных библиотек не возникло бинарной несовместимости с собранными XS-модулями. К тому же ожидание процесса завершения компиляции и тестов на каждом сервере может занимать неприлично долгое время.

Упаковка модуля в пакет того типа, который применяется в текущей системе, даёт следующие преимущества:

- фиксируется версия модуля,
- уменьшается время установки за счёт того, что компиляция и тесты происходят только во время сборки пакета,

- фиксируются все зависимости на другие пакеты, включая зависимости на сторонние библиотеки,
- отслеживаются файловые конфликты,
- отслеживаются бинарные конфликты (нельзя обновить библиотеку с новым soname без удаления всех пакетов, зависящих от старой версии),
- отслеживается целостность (md5sum, gpg),
- возможность простого обновления до новой версии и отката на старую версию.

Единственный недостаток упаковки — это жёсткая привязка к той платформе, на которой работает данный менеджер пакетов. Но если в эксплуатации отсутствует зоопарк дистрибутивов, то этот недостаток

незаметен.

dh-make-perl

В составе дистрибутива Debian присутствует утилита `dh-make-perl`, которая значительно упрощает процесс создания deb-пакета для Perl-модулей. В большинстве случаев практически не требуется вникать в суть того, как организованы deb-пакеты, как происходит их сборка, что очень удобно для начинающих.

Для её установки достаточно вызывать команду:

```
$ sudo apt-get install dh-make-perl
```

Эта команда, как и все последующие в

данной статье примеры команд, выполняются на системе *Ubuntu Server 12.04.3 LTS*. К сожалению, идущий в составе этого дистрибутива `dh-make-perl 0.75` сломан и мало пригоден к работе, поэтому первым делом его требуется обновить до версии `0.79-1`, которая есть в репозитории *Ubuntu Saucy*. Достаточно вручную загрузить пакет с launchpad.net и установить вручную с помощью `dpkg`:

```
$ sudo dpkg -i dh-make-perl_0.79-1_all.deb
```

Также будет полезным утилиты для сборки пакетов `devscripts` и модуль `local::lib`:

```
$ sudo apt-get install devscripts liblocal-lib-perl
```

Первое, о чём нужно упомянуть, — это

способность `dh-make-perl` искать имя пакета в репозитории по имени модуля. Например, чтобы узнать в каком пакете находится модуль `File::ShareDir` надо выполнить команду:

```
$ dh-make-perl locate File::ShareDir
...
File::ShareDir is in libfile-sharedir-perl package
```

Таким образом, модуль `File::ShareDir` содержится в пакете libfile-sharedir-perl. По принятой традиции в Debian имена пакетов для Perl-дистрибутивов всегда пишутся с маленькой буквы, и полученное имя оборачивается в формат `lib%s-perl`. Т.о. по имени модуля можно догадаться о названии его пакета, но угадать получается не всегда, например:

```
$ dh-make-perl locate Test::Git
```

```
...
Test::Git is in libgit-repository-perl package
```

Для поиска по именам модулей `dh-make-perl` использует утилиту `apt-file`. Поэтому перед выполнением поиска необходимо установить `apt-file` и обновить локальную копию базы данных файлов в подключённых репозиториях:

```
$ sudo apt-file update
```

`dh-make-perl` может самостоятельно искать и загружать пакеты со CPAN, для этого он использует классическую утилиту `cpan` из одноимённого модуля. Поэтому при первом использовании `dh-make-perl` скорее всего придётся ответить на вопросы конфигурации модуля `CPAN` о поиске зеркала и т.д.

Сборка Path::Tiny

Для примера попробуем собрать модуль `Path::Tiny`. Нацелим `dh-make-perl` на CPAN и попробуем загрузить модуль:

```
$ dh-make-perl --cpan Path::Tiny
...
Needs the following debian packages during building: libcapture-tiny-perl, libtest-failwarnings-perl, libfile-pushd-perl, perl (>= 5.13.4), libtest-fatal-perl, libdevel-hide-perl, libtest-deep-perl
Needs the following modules for which there are no debian packages available:
 - Test::FailWarnings
```

Утилита загрузила последнюю версию модуля со CPAN, проанализировала его зависимости и выдала очень полезную информацию о том, что нам потребуется для сборки модуля. Прежде всего указан список зависимостей пакетов, которые уже доступны в репозиториях Ubuntu, и их можно установить простым `apt-get install`. Также указан и модуль `Test::FailWarnings`, которого нет в репозиториях. Таким образом, перед тем как продолжить, необходимо собрать модуль `Test::FailWarnings`.

```
$ dh-make-perl --cpan Test::FailWarnings
...
Needs the following debian packages during building: libcapture-tiny-perl (>= 0.12), perl (>= 5.13.4)
```

Данный модуль может быть собран, для него требуется установить лишь `libcapture-tiny-perl`:

```
$ sudo apt-get install libcapture
    -tiny-perl
```

Теперь попробуем собрать пакет:

```
$ dh-make-perl --build --cpan
    Test::FailWarnings
```

Если всё прошло благополучно, то в текущем каталоге вы получите файл `libtest-failwarnings-perl_0.006-1_all.deb`, который можно сразу установить в систему:

```
$ sudo dpkg -i libtest-
    failwarnings-perl_0.006-1_all.
    deb
```

Установим и другие зависимости `Path::Tiny` из репозитория:

```
$ sudo apt-get install libfile-pushd-perl libtest-fatal-perl \
    libdevel-hide-perl libtest-deep-perl
```

Попробуем собрать `Path::Tiny`:

```
$ dh-make-perl --build --cpan Path::Tiny
...
Warning: prerequisite File::Spec 3.40 not found. We have 3.33.
Warning: prerequisite autodie::exception 2.14 not found. We have 2.1001.
```

Сборка завершится ошибкой, поскольку два необходимых нам модуля `File::Spec` и `autodie::exception` устаревшие, и их

требуется обновить.

Оба этих модуля входят в состав базового Perl, поэтому `dh-make-perl` не даст их собрать:

```
$ dh-make-perl --build --cpan
    autodie::exception
 autodie::exception is a standard
     module. Will not build
    without --core-ok.
```

Но он сразу подскажет, что надо сделать, чтобы сборка прошла успешно: указать опцию `--core-ok`:

```
$ dh-make-perl --core-ok --build
    --cpan autodie::exception
$ dh-make-perl --core-ok --build
    --cpan File::Spec
```

Установим получившиеся пакеты:

```
$ sudo dpkg -i libautodie-perl_2.20-1_all.deb \
    libpathtools-perl_3.40-1_amd64.deb
```

После чего сборка `Path::Tiny` завершится успешно, и можно будет установить полученный пакет с помощью `dpkg`.

Репозиторий пакетов

Со временем количество собранных пакетов будет расти, и установка их по отдельности превратится в кошмар. Поэтому лучше всего сразу организовать свой apt-репозиторий для собранных пакетов, из которого можно будет централизованно устанавливать все пакеты.

Для этих целей в Debian существует утилита `reprepro`, позволяющая организовать собственный репозиторий пакетов, пригодный для использования наряду с официальными репозиториями дистрибутива. Устанавливается утилита командой:

```
$ sudo apt-get install reprepro
```

Затем создаётся каталог, в котором будет располагаться репозиторий, например:

```
$ mkdir ~/repo
```

После чего внутри него создаётся каталог `conf`, в котором создаётся файл `distributions`, например, так:

```
$ cat <<EOF >  ~/repo/conf/
    distributions

Origin: cpan
Label: cpan
```

```
Codename: precise
Architectures: i386 amd64 source
Components: main
Contents: .gz
Description: fresh cpan packages

EOF
```

В данном файле задаётся список дистрибутивов, для которых будет создан наш репозиторий, указывается, для каких архитектур будут доступны собранные пакеты. Кроме того, есть возможность разбить пакеты на группы (компоненты), в данном примере создаётся только один компонент — `main`.

Инициализируем репозиторий:

```
$ reprepro –Vb ~/repo export
```

После этого добавим в репозиторий один

из собранных пакетов:

```
$ reprepro -b ~/repo -C main includedeb precise \
    ~/libautodie-perl_2.20-1_all.deb
```

Утилита скопирует пакет в наш репозиторий и обновит все необходимые индексные файлы. Пакет будет отнесён к дистрибутиву `precise` в составе компонента `main`.

Для того, чтобы репозиторий стал доступен для установки пакетов в систему, необходимо добавить запись о нём в `/etc/apt/sources.list`:

```
deb file:///path/to/repo precise main
```

Как видно, указывается путь к репозиторию, кодовое имя дистрибутива и список нужных компонентов.

Чтобы обновить индексы `apt`, необходимо выполнить команду:

```
$ sudo apt-get update
```

После чего пакеты, которые были добавлены в репозиторий, станут доступны для установки с помощью команды `apt-get install`. Аналогичным образом после выполнения:

```
$ sudo apt-file update
```

начнёт работать поиск по файлам в пакетах нового репозитория.

Чтобы сделать данный репозиторий доступным для других серверов, можно воспользоваться любым веб-сервером, раздающим статические файлы, и опубликовать репозиторий в его конфигурации. Например, в случае `nginx`, можно указать

конфигурацию:

```
server {
    listen 80;
    server_name mycpan.example.
       org;

    location / {
        root /path/to/repo;
    }
}
```

В таком случае в `/etc/apt/sources.list` указывается строка:

```
deb http://mycpan.example.org/
    precise main
```

Заключение

`dh-make-perl` позволяет значительно сэкономить усилия по созданию deb-пакетов, но, к сожалению, не во всех случаях удаётся легко выполнить преобразование. Процесс дальнейшей поддержки пакета, в том числе и периодическое обновление, потребует знаний о сборочных инструментах среды Debian/Ubuntu. Во всех этих случаях рекомендую ознакомиться с «Руководством начинающего разработчика Debian», которое поможет решить многие возникающие вопросы.

■ *Владимир Леттиев*

8 Обзор CPAN за август 2013 г.

Рубрика с обзором интересных новинок CPAN за прошедший месяц.

Статистика

- Новых дистрибутивов — 219
- Новых выпусков — 912

Новые модули

- Time::Limit Иногда люди пишут кривые скрипты, которые могут зависнуть или зациклиться. Если PHP-разработчиков это никогда не заботило, т.к. PHP имеет встроенные

средства для уничтожения зависших скриптов, то для Perl-разработчиков это было проблемой, и им приходилось искать причину в своих программах. Теперь появился модуль, который позволяет по таймауту автоматически уничтожать работающую слишком долго perl-программу.

- Caroline Модуль является ещё одной реализацией библиотеки для создания интерфейса командной строки и редактирования строк. В отличие от классической реализации `Term::ReadLine::Gnu`, данный модуль не имеет зависимостей от C-библиотек и также поддерживает автозавершение команд.

- Regexp::VerbalExpressions Perl-реализаци
JavaScript-библиотеки *VerbalExpressions*, которая позволяет создавать сложные

регулярные выражения, используя простой синтаксис элементарных команд, последовательно описывающих выполняемые действия. Например:

```
my $url_re = verex
    ->startOfLine->than("
       http")->maybe("s")->
       then('://')
    ->anything_but(' ')->
       end_of_line;
```

- Dist::Inkt Ещё один сборщик дистрибутивов Perl-модулей. Как описывает сам автор модуля, в `Dist::Zilla` ему не хватало нужного уровня сумасшествия, поэтому он написал эту альтернативу.

- ZMQx::Class Модуль является OO-интерфейсом к библиотеке *ZeroMQ*, управляющей передачей сообщений

между процессами через различные виды транспортных протоколов.

- Class::Tiny Невероятно, но это ещё один минималистичный модуль для создания классов. Дэвид Голден создал наиболее минималистичный вариант ООП-фреймворка, который позволяет создавать методы акцессоров и мутаторов и при этом не имеет зависимостей от модулей, которые отсутствуют в составе базового Perl. Он может быть использован вместо `Class::Struct` или других подобных самопальных ООП-движков, используемых в модулях, входящих в ядро Perl.

- Hash::Convert Любопытный модуль, который позволяет по заданным правилам изменять сложные структуры данных, добавляя новые поля, изме-

няя существующие, задавая значения по умолчанию и множество других преобразований.

- C::TinyCompiler Модуль является обвязкой к *Tiny C Compiler*, являющимся быстрым и небольшим C99-совместимым C-компилятором, имеющим возможность компилировать C-код в машинный код и запускать его без промежуточного сохранения кода на диск (JIT).

- XS::TCC Ещё один проект обвязки к *Tiny C Compiler*, по смыслу схожему с `Inline::C` — включение C-кода в Perl-код, но с возможностью выполнения его на лету без промежуточного создания исполняемого файла на диске.

- File::HashCache Модуль, который позволяет обрабатывать набор файлов

и кешировать полученный результат в виде файлов, содержащих в имени MD5-хэш исходного файла. Это может быть использовано в веб-разработке, когда перед отправкой css- или js-файлов их нужно предварительно минимизировать, сохранить промежуточный результат, а чтобы обойти проблему кеширования на прокси клиента — изменять имя файла после каждого изменения содержимого.

Обновлённые модули

- List::Util 1.31 Обновился модуль `List::Util`, в котором появился набор функций, которые работает со списками пар (массив с чётным числом

элементов), такие как `pairgrep` — для поиска пар, удовлетворяющий условию в блоке, `pairfirst`, `pairmap` и прочие. Идея подобных функций была взята из модуля `List::Pairwise`.

- MIME::Types 2.02 Вышел новый мажорный релиз `MIME::Types`. В новом релизе значительно расширена база типов и радикально увеличена скорость запуска модуля и уменьшен объём занимаемой памяти.

- Data::Alias 1.17 Обновлён модуль для создания псевдонимов переменных. В новой версии исправлено несколько ошибок и добавлена поддержка Perl >= 5.17.6.

- Module::CPANfile 1.0001 Вышел первый стабильный релиз модуля для обработки файлов в формате `cpanfile`.

В данном выпуске зафиксирована версия 1.0 спецификации формата `cpanfile`, который, вероятно, станет стандартом для описания зависимостей модулей на CPAN.

- Carton 1.0.9 Первый мажорный релиз менеджера зависимостей для Perl-приложений. Вместе с фиксацией формата `cpanfile` стабилизируется и разработка с использованием carton. Обновлены документация и руководство по использованию carton.

- Protocol::SPDY 1.000 Абстрактная реализация протокола *SPDY* для Perl достигла своего первого стабильного релиза. Реализованы новые возможности и исправлено множество ошибок в модуле.

- dh-make-perl 0.79 Обновилась утилита для создания deb-пакетов из CPAN-модулей, которые можно использовать для перепаковки архивов модулей из CPAN в модули, пригодные для установки в системах Debian/Ubuntu. В новой версии исправлены проблемы совместимости с Perl 5.18.

- Net::SSH::Perl 1.36 Выпущен новый релиз модуля реализации клиента SSH-протокола на чистом Perl. В новом выпуске исправлено несколько ошибок.

- snaked 0.14 Новый релиз менеджера задач, альтернативной реализации *cron* на Perl. Релиз примечателен тем, что был выпущен во время проведения конференции YAPC::Europe 2013 в Киеве (автор делал доклад о модуле).

- Function::Parameters 1.0211 Модуль является реализацией сигнатуры функций для Perl. Выпуск данного модуля также был сделан автором во время доклада на конференции YAPC::Europe 2013.

- Quota 1.7.0 Perl-интерфейс для доступа к квотам файловых систем. В новой версии реализована поддержка квот на системах NetBSD 6.0, а также добавлена поддержка квот более 4 Гб на 32-битных системах.

- Devel::Cover 1.08 Новый релиз модуля для измерения покрытия кода тестами теперь поддерживает Perl 5.18.1.

- Starlet 0.20 Выпущен новый релиз высокопроизводительного PSGI/Plack HTTP-сервера. В данном релизе появилась поддержка протокола HTTP версии 1.1.

- DBD::SQLite 1.40 В новом релизе DBD::SQLite добавлена поддержка функции `statistics_info`.

■ *Владимир Леттиев*

9 Интервью со Stevan Little

Stevan Little — автор ООП-фреймворков Moose и tor, альтернативного perl5 интерпретатора на Scala — Moe, и многих других Perl-модулей.

Когда начал программировать?

Первый раз я программировал на BASIC в 1984, когда моя семья купила Apple ||. В основном я программировал простую графику и безуспешно пытался программировать квесты, я потерял интерес где-то через год. Я больше не программировал до 1997 года, когда узнал про HTML и Javascript и получил работу с первыми веб-страницами.

Первоначально я просто копировал и

хачил простые Javascript-виджеты, но в конце концов купил большую книгу O'Reilly Javascript и научил себя программировать. После этого я провел несколько лет, читая книги по программированию, и выучил несколько других языков, включая такие вещи, как Ada 95, Erlang, LISP, Standard ML, Java и Python. И только в 2002 году я был нанят на текущую работу в Infinity Interactive, и я выучил Perl.

Каким текстовым редактором пользуешься?

Мне всегда больше нравились графические редакторы, до недавнего времени я был пользователем TextMate, но полгода назад я перешел на SublimeText 2. Мне он больше нравится, потому что у него есть несколько экономящих время особенностей, которыми обладают большие IDE, но он не

навязывает тебе свой порядок работы или что-то в этом роде.

Когда и как познакомился с Perl?

Как я уже сказал, выучил Perl, когда меня в 2002 наняли на текущую работу. В то время я все еще в основном занимался HTML/CSS и Javascript по работе и много упражнялся с Python в свободное время. По-началу я не был очень рад тому, что придется учить Perl, пока я не нашел книгу Damian Conway «Object Oriented Perl» и не увидел, что на самом деле может Perl. В силу того, что я изучал большое количество языков программирования, было довольно просто выучить Perl, но несколько лет потребовалось, чтобы понять, что значит писать действительно «перловый» код.

С какими другими языками приятно ра-

ботать?

Я большой поклонник языков программирования вообще, я всегда читаю и изучаю другие языки, старые и новые. Хотя я могу читать многие языки, реальные программы я писал только на семи или восьми из них.

Совсем недавно мне понравилось работать со Scala, который я использую в паре экспериментальных проектов. Я нахожу его в некоторой степени очень похожим на Perl в философии TIMTOWTDI (есть больше одного способа сделать это – *прим. перев.*). Однако у него очень глубокие академические и функциональные корни, которые позволяют сторониться от более практичной философии Perl «главное, чтобы работало». Но это все еще очень молодой язык, и очень интересно наблю-

дать, в какую сторону будет двигаться сообщество.

На работе у нас большое количество проектов на C#, и мне всегда очень нравилось работать с этим языком. В каком-то смысле это «Java, сделанная правильно» в том смысле, что дизайнеры языка очевидно многое почерпнули из Java и знали как не надо делать. Также хочу отметить, что уделять внимание основным библиотекам, как, например, в C#, очень полезно, многие open source языки не думают об этом, что усложняет добавление их в будущем.

Также я недавно игрался с TypeScript, попыткой Microsoft сделать Javascript более подходящим для больших задач. Мне понравился этот язык, жду, когда выйдет версия 1.0.

Какое, по-твоему, самое большое преимущество Perl?

По-моему, самое большое преимущество, как и самый большой недостаток, это философия TIMTOWTDI.

Из всех языков, которые я изучал и с которыми работал, ни один из них не дотягивает до гибкости Perl. В качестве хорошего примера можно привести новое API ключевых слов, что позволяет прервать парсер интерпретатора при исполнении и изменить синтаксис языка, мне неизвестно ничего из похожего в других языках. И это не просто новая вещь в Perl, у нас всегда был доступ к op-tree через модуль B, с помощью которого можно сделать много страшных вещей. Такой уровень доступа к runtime языка позволяет делать много интересных и полезных вещей в

Perl. Вместе с этой гибкостью, Perl также надежно ограничивает такие функции, что позволяет их безопасно использовать даже в продакшн-системах.

Но в тоже время такая гибкость и ограниченность также проблема Perl.

Perl из-за TIMTOWTDI это язык со многими диалектами; хорошими, плохими и совсем ужасными. Я убежден, что именно поэтому у Perl репутация неподдерживаемого языка. У любого проекта с большим количеством кода на любом языке будет свой внутренний диалект, созданный в течение длительного времени его автором(-ами), и если повезет, это может помочь новому программисту в чтении и понимании кода (когда он выучит диалект, конечно). Для более строгих языков типа Python или Java вариации диалектов, со-

существующих в одном коде, довольно мало. Но такой гибкий язык как Perl, с его исключительно искушенным парсером, способен поддерживать большое количество диалектов одновременно. Это сильно усложняет жизнь новому программисту для обучения, поддержки и безопасного расширения кода.

Какими, по-твоему, особенностями должны обладать языки будущего?

Мне кажется, что хорошая модель распараллеливания будет самой важной частью любого языка будущего. Я лично предпочитаю модель акторов с полным разделением, потому что я нахожу ее наиболее практичной и самой простой. Она тестируется в бою языком Erlang в течение уже 20 лет и в последнее время перенимается Java- и Scala-сообществами с большим успехом. Я

не говорю, что модель акторов это лучшая модель, но она наиболее интересная в данный момент.

Также мне кажется, что гибкий и надежный вывод типов тоже будет ключевым свойством. В прошлом вывод типов можно было найти только в таких функциональных языках, как Standard ML, OCaml и Haskell со строгой типизацией, но недавно его начали применять и в динамических языках. Примером может служить компилятор TypeScript, который, получив на входе обыкновенный Javascript (так как TypeScript это расширение Javascript), проверит все типы, даже без их указания. Это заставляет программиста быть более строгим при использовании переменных — если переменной присвоено целое число, нельзя затем присвоить строку — TypeScript также предоставляет обходной

путь в виде типа “Any”, который позволяет программисту нарушать это правило. Такая комбинация гибкости и жесткости, как мне кажется, будет хорошим свойством любого языка будущего.

И я думаю, что любой язык будущего должен иметь надежную объектную систему с мощными возможностями мета-программирования.

Почему написал Moose? Почему он стал таким популярным?

Я написал Moose после того как поработал с объектной системой Perl 6 в рамках моей работы над проектом Pugs. После длительной работы с хорошей объектной системой Perl 6 возвращение к базовой OO в Perl 5 очень сильно показало, какой муторной и скучной она может быть. Я уже прототипи-

ровал объектную систему Perl 6 четыре или пять раз для Pugs, поэтому я просто перенаправил свои усилия, чтобы сделать что-нибудь работающее и для Perl 5. Были две или три неудачные попытки, прежде чем я написал Class::MOP и затем на его основе написал Moose.

Я думаю, что Moose стал популярным, потому что люди хотели хорошей объектной системы Perl 6 и с надеждой ждали ее в течение несколько лет. У Moose получилось реализовать это желание, и после этого он быстро распространился, потому что эти люди рассказывали о Moose другим.

Что думаешь об альтернативных или, так называемых «легких», реализациях Moose?

У меня нет с этим никаких проблем, они

заполняют нишу, которую Moose никогда не заполнит. В течение последних нескольких лет было много таких реализаций, хороших и плохих, сейчас я думаю, что наилучшим является Moo, потому что он позволяет прозрачно «проапгрейдиться» до Moose, если потребуется.

Что за проекты mop и mop-redux?

Проект p5-mop был первой попыткой написать объектную систему, похожей на Moose, которая бы могла войти в ядро Perl. В конечном счете он был раздавлен весом собственной сложности и несколькими хитрыми багами, которые невозможно было обойти. Должен признаться, что этот факт привел меня в уныние, и в течение некоторого времени я начал негативно относиться к ядру Perl. Это, в свою очередь, привело меня к проекту Moe — попытке

написать Perl 5-подобный язык, используя Scala. Moe превратилось в эксперимент с концепциями языка, которые я бы хотел видеть в Perl 5, но которые сложно прототипировать с помощью самого Perl 5.

В самолете домой после YACP::NA в этом году я решил дать p5-mop второй шанс, на этот раз воспользовавшись опытом, который я приобрел при разработке Moe. Это привело к проекту p5-mop-redux, над которым я сейчас работаю. У меня большие надежды на него, и я буду им заниматься несколько следующих месяцев и буду пробовать проталкивать в ядро Perl.

Может ли использование ООП-фреймворка без понимания привести к плохим практикам программирования? Возможно ли писать «чистый код» без ООП-фреймворка в Perl?

Мне кажется, что плохое понимание своих инструментов часто приводит к плохому их использованию, не думаю, что ООП-фреймворк чем-то в этом случае отличается. Вполне возможно написать чистый код без ООП-фреймворка, код Plack может служить отличным примером этому.

Не разделен ли, по-твоему, сейчас CPAN на два лагеря: те, кто используют Mo*, и другие?

Нет, мне кажется, что разделение больше между «Зависимости — это хорошо» и «Зависимости — это зло». Пользователи Mo* больше попадают в лагерь «зависимости это хорошая вещь», остальные в «зависимости это плохо», и я не думаю, что Mo* является здесь разделительной чертой. Надеюсь, что Moo, имея небольшое количество зависимостей, поможет построить

мост между этими двумя лагерями.

Надеюсь также, что это исправится и проектом p5-mop-redux. Если у нас будет расширяемая объектная система в ядре, тогда люди не будут беспокоиться о бремени зависимостей, когда им нужны объекты.

Где сейчас работаешь? Сколько времени уделяешь программированию?

Я работаю в небольшой консультационной фирме Infinity Interactive, у нас 24 постоянных сотрудника и несколько постоянных фрилансеров. Мы предоставляем разные сервисы, такие как Perl-программирование, высококачественная HTML/CSS-верстка, сложные пользовательские интерфейсы на Javascript, .Net- и Java-программирование, консультирование в сфере системного администрирования и многое другое. На

данный момент в основном я занимаюсь управлением и дизайном архитектуры проектов больше, чем программированием для клиентов, но не все так плохо, ведь у меня больше времени заниматься open source проектами, в том числе и p5-mop.

Стоит ли советовать молодым программистам изучать Perl?

Да, но со временем. Я так говорю, потому что не думаю, что Perl это хороший язык для изучения программирования. Мне кажется, что Java или Python больше подходят для этого. Однако, я уверен, что в какой-то момент в своей карьере стоит попробовать выучить Perl, потому что это сильно расширяет кругозор программиста, который хочет учиться. Я не встречал другого языка, похожего на Perl, изучение и понимание которого сделает из тебя

лучшего программиста.

■ *Вячеслав Тихановский*

10 Perl Quiz

Perl Quiz — уже ставшая традиционной на многих Perl-конференциях викторина на «знание» Perl. Почему в кавычках? Это вы поймете из самих вопросов. Ответы на викторину в текущем выпуске будут опубликованы в следующем. Итак, поехали!

Ответы из предыдущего выпуска: 1) 3, 2) 4, 3) 4, 4) 3, 5) 2, 6) 3, 7) 3, 8) 4, 9) 3, 10) 3.

1. Что Ларри Уолл предлагает позаимствовать из Perl 6 в Perl 5?

    1. Долгую и обстоятельную разработку.
    2. Создание многочисленных форков компилятора.

    3. Считать код программы по умолчанию написанным в юникоде.
    4. Оператор `if` с двумя условиями (`if $a < $x < $b`).

2. Что будет изображено на новом сайте DBIx::Class?

    1. Пирамиды, пальмы и верблюды.
    2. Реки, озера и моря.
    3. Леса, соль и лоси.
    4. Книги, жесткий диск и карандаш.

3. На какой сайт модуль CPAN::Reporter отправляет отчеты о тестировании модулей?

    1. cpan.org
    2. cpantesters.org
    3. metacpan.org
    4. perlresume.org

4. Какая страница на Фейсбуке является официальной страницей YAPC Europe Foundation?

    1. facebook.com/yapceu
    2. facebook.com/yapceurope
    3. facebook.com/pages/YEF-YAPCEurope Foundation/145046488917748
    4. facebook.com/pages/Yet-Another-Perl-Conference/145177855492256

5. Какой текст на футболках YAPC::Europe 2013 выбрал только один человек?

    1. Camelia Perl
    2. Rakudo Perl
    3. Perl 6
    4. Perl 7

6. Какое настоящее имя у Zefram’a?

    1. Lukáš Rampa

2. Stevan Little
3. Tim Toady
4. Andrew Main

7. Какой будет тема конференции YAPC::Europe 2014?

    1. Enlightened Perl
    2. Distilled Perl
    3. Distributed Perl
    4. Disassembled Perl

8. Что нового появилось недавно на CPAN для удобного ООП в перле?

    1. Moose::Tiny
    2. Object::Tiny
    3. Class::Tiny
    4. Tiny::Tiny

9. На какой конференции в последние годы собирается больше участников?

1. YAPC::Asia
2. YAPC::Brasil
3. YAPC::Europe
4. YAPC::NA

10. Для чего создатели предлагают использовать будущий сайт PRForge.com?

    1. Быть кузницей кадров для компаний, использующих Perl.
    2. Конкурировать с SourceForge.com.
    3. Устроить краудфандинг для найма разработчиков ядра языка.
    4. Автоматически ретвитить, лайкать и шарить посты про Perl.

■ *Андрей Шитов*