SPECIAL ОПРОС

АЛЕКСЕЙ ПЕТРОВ

В IT 20 лет. Эксперт в области защиты данных, эксперт по компьютерным преступлениям, эксперт по сетевым коммуникациям и телефонии. Сертификаты от Novell/ 3com/Bay/Siemens/Cisco/ ISACA. Консультант по вопросам ИТ-безопасности в Secproof Oy *(www.secproof.com)*. Свободный консультант Arhont.com, iPRO.lv.

КРИС КАСПЕРСКИ

Известен еще как мыщъх. Компьютеры грызет еще с тех времен, когда Правец-8Д считался крутой машиной, а дисковод с монитором были верхом мечтаний. Освоил кучу языков и операционных систем, из которых реально использует W2K, а любит FreeBSD 4.5. Живет в норе, окруженной по периметру компьютерами и стеллажами с литературой.

АНАТОЛИЙ СКОБЛОВ

Последние 17 лет — системный программист, аналитик. Работает дома на себя или на заказчиков. Из известного — ядро outpost personal firewall, модем russian courier. Сфера профессиональных интересов — безопасность, телефония, интернет и так далее.

АЛЕКСАНДР ЛОЗОВСКИЙ

Если в двух словах — этому человеку СПЕЦ обязан своей жизнью. Практически каждый номер Александр делает ему искусственное дыхание в условиях, приближенных к реальным военным действиям :). Кроме того, — редактор «Кодинг» Хакер'а.

ИВАН (SKYWRITER) КАСАТЕНКО

Возбуждающий идеи на редколлегии. Редактор диска к журналу хакерСПЕЦ. В прошлом и настоящем успешный программист. Горячий любитель .net.

СЛУЧИТСЯ ЛИ КОГДА-НИБУДЬ, ЧТО ОДИН ИЗ ЯЗЫКОВ БУКВАЛЬНО «ЗАВОЮЕТ МИР»?

АЛЕКСЕЙ ПЕТРОВ: Очень маловероятно. Утверждение подобного рода можно сравнить с попыткой заявить что «когда-нибудь проявится universal-swiss-knife, в котором будет все, и он вытеснит все ножи и инструменты с рынка». Каждый язык программирования для решения конкретной комплексной задачи имеет свои плюсы и минусы. Каждый язык программирования имеет свою направленность и специализацию. Языки бывают узкопрофильные и широкопрофильные, сложные и простые, компилируемые и интерпретируемые, близкие к аппаратной части (низкоуровневые) и аппаратно-независимые слоеные-кроссплатформенные пироги (языки высокого уровня).

Причем высокоуровневые языки от аппаратной реализации компьютера помимо множества плюсов имеют и минусы. В частности, они не позволяют создавать простые и точные инструкции к используемому оборудованию. Программы, написанные на языках высокого уровня, проще для понимания программиста, но гораздо менее эффективны, чем их аналоги, создаваемые при помощи низкоуровневых языков. Одним из следствий этого стало добавление поддержки того или иного языка низкого уровня (язык ассемблера) в большинство современных профессиональных высокоуровневых языков программирования.

ДМИТРИЙ КОВАЛЕНКО
Системный программист, разработчик в infopulse ukraine. Хобби — теоретическая вирусология, в частности, математическое моделирование полиморфных алгоритмов в вирусах.
www.vr-online.ru

МИХАИЛ ФЛЕНОВ
Внештатный автор X почти с самого рождения журнала, создатель сайта *www.vr-online.ru*, автор 11 книг на русском и 4 на английском языке.

Изначально глупо писать узкоспециализированные драйвера устройства на далеком от железа и его основ кроссплатформенном языке, таком как, скажем, Java (хотя в жизни могут найтись всякие извращенцы). В реальности вполне возможны ситуации, когда на языке «А» задачу придется кодить в 3-4, а то и в 10 раз дольше, но удастся решить рациональнее по ресурсам, а на языке «Б» ее проще будет описать, но работать будет долго и ресурсоемко. А в жизни решить задачу надо экономически выгодно, посему будет взят кодер, который знает только язык «В», на котором эта задача и будет реализована — долго, сложно, ресурсоемко, неэффективно — но зато дешево и сердито...

КРИС КАСПЕРСКИ: По данным лингвистов, недавно собравшихся на одной шумной конференции, 70% ныне используемых языков через несколько десятков лет скорее всего перейдут в разряд мертвых. И будут как греческий или латынь. Кстати, если вспоминать историю, то и греческий, и латынь оказали колоссальное влияние на большинство современных языков.

То же самое наблюдается и с языками программирования. Термин «завоевание» — очень точный. Фирмы-создатели компиляторов/фреймворков и прочих технологий вкладывают в них нехилые деньги и двигают на рынок, зачастую действуя вопреки интересам пользователей. Новых идей ни у кого нет, поэтому все языки становятся более и более похожими друг на друга. Иные концепции просто отметаются. Взять тот же forth или lisp. И где они сейчас?! Вокруг засилье Си++-подобных языков с одинаковыми парадигмами объективного программирования, планомерно эволюционирующего в метапрограммирование, которое выросло из шаблонов, а шаблоны выросли из препроцессоров.

Так что ничего радикально нового на рынке, по сути, и нет. Меняется только синтаксис и оверхид. Как говорится, усложнять легко — упрощать трудно. Чтобы изобрести простой и гибкий язык, каким является тот же Си — это же талант нужен. А добавлять в Си++ новые фичи может любой индус. Только кто этим языком будет пользоваться?! Говорят, что среднестатистический пользователь MS office использует 2% возможностей. А сколько возможностей использует средневзятый приплюснутый программист, если все 100% фич пока что не поддерживает ни один компилятор?!

На самом деле, война умов уже давно закончена и сейчас идет брожение. С другой стороны, существует такая классная штука, как язык Пролог, но под него не существует эффективных компиляторов. А не существует их потому, что не существует адекватных процессоров, но они могут появиться в любой момент! Мы не знаем технологий, которые еще не открыты, поэтому не можем сказать, каким будет мир через десять лет. Но навряд ли один язык победит остальные, поскольку даже если он появится, тут же кто-то придумает другой язык, более удобный для решения определенного круга задач, который может как расширяться, так и сужаться...

АНАТОЛИЙ СКОБЛОВ: Возможно, это будет арабский язык. Языки программирования — исключено.

АЛЕКСАНДР ЛОЗОВСКИЙ: Я думаю, произойдет следующее. Прямо скажем, друзья, произойдет апокалипсис. Энтропия вселенной достигнет критического уровня, в результате люди станут злыми, брат пойдет войной на брата, сын — на отца. С неба посыплются гигантские камни, машины восстанут против людей, разверзнутся хляби небесные, а реки станут красными от крови. Но пока что это нам не грозит. C#, может быть, и имеет шанс «завоевать» мир, но это не значит, что кроме .NET'а ничего никому не будет нужно.

ИВАН КАСАТЕНКО: Я так не думаю. Давай проведем параллель с реальным миром — на протяжении многих веков существует целое множество языков: кто-то говорит на одном, кто-то на другом. И не факт, что самый популярный из них статистически является самым удобным (взять, хотя бы, китайский). Но даже несмотря на популярность, ни один из них так и не стал общепринятым стандартом, хотя и были даже искусственные попытки его ввести.

Причин тому несколько. Каждый язык формировался согласно национальным особенностям и необходимостям, а потому наиболее удобен конкретной нации. Человеку свойственна привычка, он не хочет менять того, что было веками. С языками программирования абсолютно та же ситуация. Есть самые распространенные: C, C++, C# (стараниями Брата), но никто не отменяет существования языков совершенно другой природы — процедурных, декларативных и т.п. Каждый из них (Nemerle, Haskell, Prolog, даже COBOL) хорош по-своему, у каждого есть сторонники, а значит — каждый обречен на выживание.

ДМИТРИЙ КОВАЛЕНКО: Думаю, такого не случится. Уже сейчас в программировании столько всяких идеологий, архитектур и платформ! Ни один язык не может полностью «накрыть» все это разнообразие. И в будущем вряд ли что-то существенно поменяется.

ЧТО ЛУЧШЕ: СИ ИЛИ СИ++?

АЛЕКСЕЙ ПЕТРОВ: C++ родился из C и очень похож на родителя. Изначально C++ был препроцессором, переводящим его конструкции в код C, который уже дальше передавался компилятору. C++ включает в себя C, и даже практически полностью с ним совместим, что позволяет некоторым писать на C++ как на C, считая его просто расширением (хотя рано или поздно либо уходят в полный C++, либо спускаются обратно на C). Это также позволяет использовать в C++ старые наработки на C. Но сравнивать C++ и C — сопоставимо с извечным вопросом, что лучше: курица, гусь или яйцо. C++ и C, продолжая развиваться и влияя друг на друга, давно представляют собой хороший конгломерат здорового симбиоза. Заговоздкой языка C стало то, что он оказался слишком низкоуровневым для задач. Задача красиво решалась, но тонула и пряталась от понимания в технической реализации. Предназначением C++ было

сделать написание программ более простым и приятным, немножко подняв планку уровня С и расширив его возможности. Новые веяния требовали от С средства работы с абстрактными типами данных, объектов, что и было реализовано введением в С++ механизма классов, позволяющих определять и использовать новые типы данных на основе существующих.

Кстати, в качестве базового языка С (для С++) был выбран не случайно, потому что он:

1. МНОГОЦЕЛЕВОЙ, ЛАКОНИЧНЫЙ И ОТНОСИТЕЛЬНО НИЗКОГО УРОВНЯ.
2. ОТВЕЧАЕТ БОЛЬШИНСТВУ ТРЕБОВАНИЙ СИСТЕМНОГО ПРОГРАММИРОВАНИЯ.
3. ИДЕТ ВЕЗДЕ И НА ВСЕМ.
4. В ТОМ ЧИСЛЕ ПРИГОДЕН ДЛЯ ПРОГРАММИРОВАНИЯ НА UNIX.

КРИС КАСПЕРСКИ: Это то же самое, что сравнивать километры с литрами, хотя в каком-то смысле автомобилисты так и поступают (расход топлива). Но все-таки это разные языки, и далеко не во всех задачах оправдано использование Си++. И уж тем более не факт, что время, вложенное в его изучение, окупится ускоренной разработкой программ. Но это уже священные войны начинаются...
Си — низкоуровневый язык, далеко не все приемлют его парадигму. Си++ — нечто очень большое и сложное, плюсов у него всего два (да и те достались в наследство от Си), а вот минусов...
ИВАН КАСАТЕНКО: Лично мне более предпочтительным кажется Си. Не знаю уж, связано ли это с моей работой или просто исторически сложившаяся симпатия. Си нравится своей простотой и читабельностью хорошо написанного кода. Гораздо сложнее (мне лично) читать код на Си++. Так что если уж выбирать классы, проектирование с использованием шаблонов и т.п., то это должен быть язык поудобнее. Мне в этом плане симпатизирует Java и C#. Так что — либо Си, либо Java/C#.
ДМИТРИЙ КОВАЛЕНКО: Если вопрос только в языках, то Си++ лучше, поскольку Си является подмножеством Си++, а значит, Си++ обладает всеми возможностями Си. Если же вопрос в том, какой подход лучше — процедурный (как в Си) или объектно-ориентированный (как в Си++), то смотря какие задачи надо решать. Для больших проектов, которые делает много людей, лучше подходит объектно-ориентированный подход. В небольших проектах вполне оправдан процедурный подход.
МИХАИЛ ФЛЕНОВ: Идеальных языков не бывает, и все зависит от задачи. Если необходимо написать офисную программу с большими возможностями, то использовать Си проблематично, а разработка отнимет очень много времени. Поэтому выбор должен пасть на Си++. Если необходима маленькая и быстрая утилита, то С++ будет излишним. И в данном случае выиграет старичок Си. Желательно знать несколько разных языков и при решении определенной задачи выбирать тот, который лучше подходит в данный момент.

КАКИЕ ПЛЮСЫ И МИНУСЫ У UNIX WAY?

КРИС КАСПЕРСКИ: Сложный вопрс, в двух словах об этом и не скажешь. Если же говорить предельно кратко, то плюсы такие:

- ВОЗМОЖНОСТЬ ПОСТРОЕНИЯ СЛОЖНЫХ СИСТЕМ ИЗ ПРОСТЫХ «КИРПИЧИКОВ», КАЖДЫЙ ИЗ КОТОРЫХ МОЖЕТ БЫТЬ ЗАМЕНЕН ДРУГИМ ИЛИ МЕЖДУ ДВУМЯ КИРПИЧИКАМИ ВСТАВЛЕН ТРЕТИЙ.
- МИНИМУМ ПОВТОРНОГО ИСПОЛЬЗОВАНИЯ КОДА, ОТКРЫТЫЕ ПРОТОКОЛЫ, ЧЕТКОЕ РАЗДЕЛЕНИЕ НА УРОВНИ.

Что касается минусов, то они такие:

- ПО МЕРЕ РОСТА СИСТЕМЫ РАБОТАТЬ С НЕЙ СТАНОВИТСЯ ВСЕ ТРУДНЕЕ И ТРУДНЕЕ, ПОСКОЛЬКУ ВМЕСТО МОНОЛИТНОГО БЛОКА У НАС ИМЕЕТСЯ МНОЖЕСТВО МАЛЕНЬКИХ БЛОКОВ, ЧАСТО ОТ НЕЗАВИСИМЫХ ПОСТАВЩИКОВ, БЕЗ ЧЕТКИХ СПЕЦИФИКАЦИЙ. И ЭТО УЖЕ НЕ ПРОГРАММА ПОЛУЧАЕТСЯ, А КОНСТРУКТОР, С КОТОРЫМ БОЛЬШЕ ТРАХАЕШЬСЯ, ЧЕМ РАБОТАЕШЬ.

АНАТОЛИЙ СКОБЛОВ: Минус — менее дружелюбная среда для пользователей (по сравнению с Windows), в первую очередь из-за того, что пользователи обычно с *nix не знакомы. Плюс — наличие бесплатных *nix'ов с открытыми исходниками, с которыми можно делать все, что угодно. Если, конечно, это требуется. Все остальное — лишь религиозные споры или частности.
АЛЕКСАНДР ЛОЗОВСКИЙ: По этому вопросу лучше обратиться к статье Криса Касперски «Так ли открыты открытые исходники» (www.xakep.ru/magazine/xs/060/076/1.asp) и статье Константина Клягина «Свободу софту» (www.xakep.ru/magazine/xs/053/032/1.asp). И труды волосатого Ричарда Столлмана будут полезны.
ИВАН КАСАТЕНКО: Плюс — свобода. Все мы любим свободу, равенство, братство. Соответственно, двигатель тут, в основном, — энтузиазм. А программисты-энтузиасты, взрощенные на ниве этой са-

мой свободы, способны горы свернуть. Минус — практическая нежизнеспособность крупных проектов. Из моего опыта не припомню ни одного жизнеспособного крупного ГНУтого проекта. Кроме, пожалуй, ядра Linux. Все остальное, прямо скажем, нещадно глючит. В качестве оправдания приводят обычно бесплатность. В общем, для крупных проектов тут не хватает главного — денег и (в большей степени) ответственности. Денег, которые позволят нанять грамотных специалистов, способных управлять командой, организовывать проект и так далее. Впрочем, деньги часто инвестируют и в «свободные» проекты. А вот второй компонент — ответственность — в большей степени все-таки свойственна коммерческим компаниям.

КАКИЕ ОСНОВНЫЕ ТЕНДЕНЦИИ СЕЙЧАС В ПРОГРАММИРОВАНИИ?

КРИС КАСПЕРСКИ: 1 АУТСОРТИНГ В СЛАБОРАЗВИТЫЕ СТРАНЫ.
2 ПРОЕКТИРУЮТ НЕ ИНЖЕНЕРЫ, А МАРКЕТОЛОГИ.
3 ТТХ НЕ ИМЕЮТ ЗНАЧЕНИЯ, ГЛАВНОЕ — ЦВЕТ.
4 ВАВИЛОНСКАЯ БАШНЯ ВСЕ ВЫШЕ И ВЫШЕ, ЗАЧЕМ — НЕПОНЯТНО, НО ВЫШЕ.
5 ДУМАТЬ НЕ НАДО, НАДО КОДИТЬ.

ДМИТРИЙ КОВАЛЕНКО: Виртуализация. Сейчас столько расплодилось всяких кроссплатформенных виртуальных машин, интерпретаторов, сред, поддерживающих скриптовые языки, data-driven технологий и прочей дряни, что разработчики почти не пишут живого машинного кода.

МИХАИЛ ФЛЕНОВ: Все движется в сторону компонентности и визуальности. Еще лет 8 назад я написал статью, в которой описывал историю языков программирования. Эта статья еще вошла в книгу «Библия Delphi». Там я говорил, что в ближайшее время победит компонентность, она станет основной технологией, что мы и увидели в последние годы (технологии Java, .NET и язык программирования Delphi — яркие представители компонентного программирования). Но вот что будет дальше, я пока сказать не могу. Следующего яркого рывка пока не вижу.

Достигли ли мы предела? Не знаю и не уверен. Когда в 93 году программировал на объектах, то думал, что это предел совершенства, — но нет, появились компоненты, которые удобнее и проще. Возможно, что на первый план выйдут web-программы, и мы уже не будем запускать на своем компьютере приложения для решения каких-либо задач. Если нужна будет офисная программа, то просто заходим на определенный сайт и работаем с нужной программой через браузер. Офисные web-программы уже есть у MS и Google. Но для того, чтобы они завоевали мир, необходима тотальная халява и высокая скорость интернета.

КАК НАПИСАТЬ БЕЗОПАСНЫЙ КОД?

АЛЕКСЕЙ ПЕТРОВ: 1 ПРОЧИТАТЬ И ИЗУЧИТЬ FAQ ПО SECURITY КОНКРЕТНОГО ЯЗЫКА ПРОГРАММИРОВАНИЯ, А ТАКЖЕ ПО БЕЗОПАСНОСТИ СРЕДЫ, В КОТОРОЙ ПРОГРАММЕ ПРИДЕТСЯ РАБОТАТЬ, И ПО БЕЗОПАСНОСТИ СОПУТСТВУЮЩИХ ПРИЛОЖЕНИЙ.

2 УДЕЛЯТЬ БОЛЬШЕ ВНИМАНИЯ ВСЕМ ВНЕШНИМ ПАРАМЕТРАМ, ПРОПУСКАЯ ИХ ЧЕРЕЗ ФУНКЦИЮ-СТЕРИЛИЗАТОР.

3 ПРОДУМАТЬ ВСЮ СТРУКТУРУ ПРОГРАММЫ И ПРОРАБОТАТЬ ОПАСНЫЕ МЕСТА.

Как правило, программисты, которые досконально знают возможности и нюансы конкретного языка и сопутствующих приложений (скажем, SQL/WEB), просто предугадывают возможные опасности и ошибки. И заранее при кодировании либо решают их, либо обходят «тонкий лед».

КРИС КАСПЕРСКИ: Во-первых, исключить всю избыточность, убрать ненужный функционал. Во-вторых, писать вдумчиво, а не на скорую руку. Но, в принципе, ответа на этот вопрос никто не знает, хоть ежегодно и выходит много книг по этой теме и появляются инструменты, призванные обезопасить программиста от себя самого, но... ошибки все равно идут косяками, программы глючат, хакеры радуются, а все почему?!

Потому что уровень культуры программирования неуклонно падает: программированию нельзя научиться по наитию и уж тем более нельзя допускать к серьезным проектам новичков, которые окончили курсы, написали несколько программ и все. Кто не умеет писать опасный код, тот никогда не сможет написать безопасный, поскольку компилятор для него — черный ящик, и он не знает, как хакеры атакуют программы.

АНАТОЛИЙ СКОБЛОВ: Любой код — безопасный, пока им пользуется небольшая группа людей. Верно и обратное — как ни старайся, ошибки будут всегда. И чем популярней твой продукт, тем больше найдется дыр. А конкретные техники написания «более безопасного» кода общеизвестны.

АЛЕКСАНДР ЛОЗОВСКИЙ: Читать книжки, искать в интернете, общаться с умными людьми, читать наш журнал :), а потом — взять, сесть и написать!

МИХАИЛ ФЛЕНОВ: Невозможно. Везде есть ошибки, поэтому тестирование, внимательность, тестирование, внимательность, тестирование, внимательность и жесткое соблюдение основных правил могут только снизить вероятность и количество ошибок. Программы пишут люди, которым свойственно ошибаться C