Bjarne Stroustrup

New York, USA.

LVEE: Tell us something about your first experience with the open-source software. Perhaps the earlier experience took place in times of large computers when being open source was typical. But it would be also interesting to know something about your first experience with free/libre licensed software (something ideologically positioned in term of freedom). Maybe it was GCC, but who knows :)

BS: I suspect my first encounter was at the first USENIX C++ conference in Santa Fe when Michael Tiemann gave a talk about how his new compiler GCC was going to do every conceivable compiler trick and soon put all other compilers out of business with his new G++ dialect. I was somewhat amused and GCC did eventually become a major factor in the C++ world, but it was several years before I could accept GCC as a C++ compiler. Before 2.95, it simply couldn't compile any program I really cared about. The early versions had many features missing and several peculiar extensions.

I am not keen on GPL because I don't see all software becoming open-source. For starters there is a lot of software that is too complex and too specialized to attract maintainers. I prefer – and use – the more liberal MIT license. The GPL library license that Michael Tiemann created to make it possible for non-ideologically-motivated organizations to use GCC and its standard library commercially was a major improvement. I think that without the LGPL, GCC and C++ would have been far less of an asset to the greater Linux community.

Generally, I try to stay out of political discussions.

L: You have already mentioned in some of your previous interviews that you prefer the diversity of the operating systems and compilers.

BS: I strongly prefer diversity of operating systems, languages, and implementations. That is partly because I fear monocultures. I wouldn't even like my favorite OS or programming language to become so dominant that the people who controlled it could become a burden on the community and/or a single point of failure. Having multiple suppliers of a technology is a good defense against stagnation.

Bjarne Stroustrup

New York, USA.

L: Расскажите, как вы впервые столкнулись с программным обеспечением с открытым исходным кодом. Наверное, самый ранний опыт пришёлся на времена больших ЭВМ, когда быть open source считалось в порядке вещей. Но было бы также интересно узнать о вашей первой встрече с ПО под свободной лицензией (нечто, идеологически позиционировавшееся в плане свободы). Возможно, это был GCC, но кто знает...

BS: Подозреваю, я впервые столкнулся с этим на первой конференции USENIX C++ в Санта-Фе, когда Майкл Тименн делал доклад о том, как его новый компилятор GCC должен был реализовать все мыслимые трюки компиляции и вскорости вытеснить все другие компиляторы с его новым диалектом G++. Меня это до некоторой степени повеселило, но GCC в конце концов стал таки главным фактором в мире C++, правда прошло несколько лет, прежде чем я смог принять GCC в качестве C++-компилятора. До версии 2.95 он просто был не в состоянии скомпилировать те программы, которые имели для меня значение. В ранних версиях не хватало многих возможностей и не было нескольких специфических расширений.

Я не особенно заинтересован в GPL, так как не считаю, что всё программное обеспечение должно иметь открытый исходный код. Для начала многие программы слишком сложны и слишком специальны, чтобы привлекать мэйнтэйнеров. Я предпочитаю – и использую – более либеральную лицензию MIT. Лицензия GPL library, которую Майкл Тименн создал чтобы организации, не обладающие идеологической мотивацией, могли пользоваться GCC и его стандартной библиотекой, стала важным улучшением. Думаю, без LGPL, роль GCC и C++ в широких кругах Linux-сообщества была бы куда менее заметной.

В целом, я стараюсь держаться в стороне от политических дискуссий.

L: Вы упоминали в одном из ваших прежних интервью, что предпочитаете разнообразие операционных систем и компиляторов. Как конференция, Linux Vacation / Eastern

For example, the competition from Clang has vastly improved GCC and Microsoft's C++ compiler.

L: As a conference, Linux Vacation / Eastern Europe is dedicated to a wide range of open source technologies, but it includes "Linux" in its name, so one of the most obvious questions is the following: which GNU/Linux distros have you mostly used? Maybe there is some distro which is more convenient for your working habits now (also, it would be interesting to know about your current usage of other Unix and Unix-like systems, if any).

BS: At work, I use Red Hat Linux. On my (Windows) Laptop, I have Ubuntu. My academic computer runs CentOS. I try hard to stay portable and to think of them all as Unix.

L: Beside Unix, you've previously worked a lot in Plan9 OS (which is something esoteric for absolute majority of our participants), so it is interesting, how would you characterize Plan9 GUI from the contemporary point of view.

BS: I was merely a – happy – user of Plan9. I did not develop for it. I still consider "Sam" the nicest editor I have ever used.

L: Can you briefly name features of the Sam editor which you liked? The fact that you actively used Sam is mentioned in Wikipedia, but there without any details about things you liked it for.

BS: It was simple, very easy to use, and fast. I mostly liked what it didn't have: no modes or complicated commands, just cut, paste, snarf, and find. A true WYSIWYG design. The handywork of Rob Pike, of course.

Unix is 50 this year; how time flies! It appears that I was one of the first four Unix users in Europe.

L: Fantastic! And who were the other three? Was it some laboratory, or so?

BS: I was visiting the computer lab of Queen Mary College in London. One of the junior lectures, Mike Cole, brought back a tape from the US with a new operating system for our small PDP11s. That was Unix. I learned a lot by browsing through the source and trying out a few things. It was so much more elegant than what I had seen before. That was in the winter/spring of 1975. About 25 years later, I bumped into Peter Salus who had just finished his book "A Quarter Century of UNIX". I read that book – I still have it somewhere – and to my

Europe посвящена OpenSource-технологиям в широком смысле, но в названии всё-таки есть слово «Linux», и поэтому один из самых очевидных вопросов – какими дистрибутивами GNU/Linux вы в основном пользовались? Может быть, есть какой-то дистрибутив, который удобнее других в плане ваших нынешних рабочих привычек? (а еще было бы интересно узнать, пользуетесь ли вы сейчас какими-то другими Unix и Unix-подобными системами).

BS: На работе я использую Red Hat Linux. На моём Windows-ноутбуке – Ubuntu. Мой компьютер в университете работает под CentOS. Я очень стараюсь оставаться мобильным и думать обо всех них как о Unix.

L: Кроме Unix, вы прежде много работали в Plan9. Для абсолютного большинства наших участников это очень эзотерическая система, поэтому было бы интересно как вы оцениваете GUI Plan9 с современной точки зрения.

BS: Я был просто счастливым пользователем Plan9, я не занимался разработкой ПО для неё. И я всё ещё считаю "Sam" лучшим текстовым редактором из всех, что я когда-либо использовал.

L: А вы не могли бы назвать, что именно вам нравилось в Sam? О том, что вы его активно использовали, упоминается в Википедии, но без каких-либо подробностей.

BS: Он был простым, очень легким в использовании и быстрым. Больше всего мне нравилось то, чего в нём не было: ни режимов, ни сложных команд, просто cut, paste, snarf и поиск. Настоящий WYSIWYG-подход. Творение Роба Пайка, разумеется.

Unix в этом году исполняется 50 лет; как время летит! Похоже, я был одним из первых четырёх пользователей Unix в Европе.

L: Фантастика! А кем были трое остальных? Это было в какой-то лаборатории?

BS: Я посещал компьютерную лабораторию Колледжа Королевы Марии в Лондоне. Один из младших лекторов, Майк Коул, привёз из США ленту с новой операционной системой для наших маленьких PDP11. Это был Unix. Я многому научился, просматривая его исходный код и пробуя разные вещи. Он был настолько элегантнее, чем всё что мне доводилось видеть прежде. Это была зима/весна 1975. Около 25 лет позже я наткнулся на Питера Салуса, только что закончившего свою книгу "A Quarter Century of UNIX". Я её прочитал – кстати, она всё ещё у меня где-то есть – и с удивлением

surprise found that Mike Cole brought the first Unix to Europe. I don't remember the names of the other two students.

L: Coming back to nowadays. What about your nowadays experience with smartphones? Your older quotation about the complexity of using modern telephones is loved by a lot of people, so it would be really interesting to know your attitude to Android or iOS.

BS: Again, I'm more a user than a developer of "smart phones". As a user of any gadget or system, I prefer for the operating system and programming languages not to be noticeable. Modern phones are insanely complicated and dangerously insecure. I really dislike when organizations try to make a system less than a computer, e.g., by interfering with the notion of a file system where I can store my information.

L: Do you feel more comfortable with integrated development environments now, or CLI compilers and text editors?

BS: I'm comfortable with good IDEs. They can significantly improve my productivity by eliminating annoyances and distractions. So, I use an IDE when I can, for example Visual Studio is very nice for C++ on Windows. I also still work from the command line when I have to – in some environments that's simpler. It is often simpler to get started from the command line as compared to learning the intricacies of an IDE.

L: Is there any GNU/Linux IDE good enough from your point of view?

BS: I tend to use the command line on Linux. That doesn't mean that isn't a good IDE that I would like, I just haven't tried one lately.

L: What about other development tools (e.g. version control systems?)

BS: We use github and the MIT license for the C++ core guidelines (`https://github.com/isocpp/CppCoreGuidelines`) and its small support library, GSL (`https://github.com/microsoft/gsl`).

L: Can you say something about your experience with the community?

BS: I obviously interact a lot with people in the C++ community, and many (probably most) of those are involved with open-source software, both as consumers and contributors. My friend and former colleague at Texas A&M University, Gabriel Dos Reis, was the shipping manager for GCC for years (I financed that). We contributed the IPR (`https://github.com/GabrielDosReis/ipr`) described in

обнаружил, что Майк Коул впервые привёз Unix в Европу. А имён остальных двух студентов не помню.

L: Что же, вернёмся в нынешнее время. Что по поводу вашего нынешнего опыта со смартфонами? Многие любят вашу прежнюю цитату о сложности современных телефонов, поэтому будет интересно узнать о вашем отношении к Android и iOS.

BS: Опять-таки, я больше пользователь чем разработчик "умных телефонов". И как пользователь любого гаджета или системы, я предпочитаю, чтобы операционная система и языки программирования оставались незаметными. Современные телефоны безумно усложнены и имеют опасно слабую защиту. Мне реально не нравится, когда организации пытаются сделать систему чем-то меньшим, чем компьютер – например, вмешиваясь в отображение файловой системы, где я мог бы хранить мою информацию.

L: Спасибо. А еще: вы сейчас комфортнее себя чувствуете с интегрированными средами разработки (IDE), или предпочитает компиляторы с интерфейсом командной строки и текстовые редакторы?

BS: Мне комфортно в хороших IDE. Они могут существенно повысить мою производительность, избавляя от некоторых надоедливых нюансов и отвлекающих факторов. Поэтому, я пользуюсь IDE когда могу, например Visual Studio очень неплохо подходит для C++ под Windows. И я всё ещё работаю в командной строке, когда приходится – в некоторых более простых окружениях. И часто бывает проще начать с командной строки, в сравнении с изучением хитростей IDE.

L: А в GNU/Linux есть достаточно хорошие IDE, как на ваш взгляд?

BS: В Linux я больше использую командную строку. Это не значит, что там нет хороших IDE, которые мне понравились бы, просто в последнее время я ничего такого не пробовал.

L: Как насчёт других средств разработки (например, системы контроля версий?)

BS: Мы используем github и лицензию MIT для C++ core guidelines (https://github.com/isocpp/CppCoreGuidelines) и его небольшой библиотеки поддержки, GSL (`https://github.com/microsoft/gsl`).

L: Можете сказать что-нибудь о своём опыте общения с комьюнити?

http://www.stroustrup.com/gdr-bs-macis09.pdf and it has been quite influential.

L: Do you think the IT sphere is affected by the uneven technologies development in different regions (when some countries are at the forefront of progress, while others are far behind)? Open-source technologies (not only software but open hardware, etc.) are supposed to be useful for such developing countries.

BS: Unless you have a monoculture keeping everyone back, you'll have uneven development. Some countries invest in education and technologies, some don't – and often fall behind. If it wasn't for technology transfer (open-source, outsourcing, whatever) developing countries would be left far behind.

L: How would you characterize students nowadays? Did they undergo any substantial changes, e.g. during the last decade or so?

BS: That's a tough question. I'm not a professional educator – I have taught a lot, even been a full-time professor, but teaching has never been my main focus. I teach because it is necessary to get my ideas used at scale. You can't just invent and design things – you have to make what you create accessible to its intended users.

There are so many more students today than when I was a student. Maybe 100 times as many computer science and IT students, maybe 1000 times. I don't see dramatic changes over the last decade, but sometime in the 1990s a change happened. I can't characterize it precisely, but that was when computer science became a mass movement worldwide, so nobody can know it all. Whereas earlier, many people in computing were idealistic "dreamers" focused on the technology, during the dot-com boom computer science came to be seen as a "get rich quick" scheme. People without deep interest in technology flocked to it. Then, after the 2002 dot-com bust, computing became to be seen as a "failed get rich quick scheme." We are now back into boom time where people flock to computers as a means to good careers and good livelihoods. There is nothing wrong with that – hard-working bright students deserve good careers – but I miss some of the idealism. Maybe there are numerically more idealists interested in pure science and technology than there were then, but they are drowned out by huge number of people looking for a safe berth. I think it sad when

BS: Конечно я много взаимодействую с людьми в сообществе C++, и многие (вероятно большинство) вовлечены в разработку ПО с открытым исходным кодом, и как потребители, и как контрибьютеры. Мой друг и прежний коллега по Университету TAMU Габриэль Дос Рейс годами был релиз-менеджером GCC (а я это финансировал). Мы контрибьютили в IPR (https://github.com/GabrielDosReis/ipr), описанный в http://www.stroustrup.com/gdr-bs-macis09.pdf, что тоже имело существенное влияние.

L: Как, по вашему мнению, влияет на сферу IT неравномерное развитие технологий в разных регионах (когда некоторые страны находятся на острие прогресса, а другие сильно отстают)? Считается, что OpenSource-технологии (не только программные, но также открытое аппаратное обеспечение и др.) полезны таким развивающимся странам.

BS: Если только вы не под властью моно-культуры, сдерживающей всех, у вас будет неравномерное развитие. Некоторые страны инвестируют в образование и технологии, некоторые нет – и поэтому они часто остаются позади. И если бы не трансфер технологий (open-source, аутсорсинг, еще что-нибудь), развивающиеся страны были бы далеко позади.

L: А как вы охарактеризуете современных студентов? Они как-то изменились, например, за последние лет десять?

BS: Это сложный вопрос. Я не профессиональный преподаватель – я много преподавал, и даже был на полной ставке профессора, но преподавание никогда не было моей главной целью. Преподавание необходимо мне чтобы получить масштаб для своих идей. Вы не можете просто изобретать и разрабатывать вещи – вам приходится ещё делать то, что вы создали, доступным для целевой пользовательской аудитории.

Сейчас студентов намного больше, чем в те времена, когда я был студентом сам. Может быть в 100 раз больше студентов Computer Science и IT, а может и в 1000 раз. Не вижу сильных перемен за последнюю декаду, но где-то в 1990-х такое изменение действительно наблюдалось. Не могу точно его описать, но это было тогда, когда направление Computer Science стало массовым во всём мире, и поэтому никто не мог знать его всё целиком. Раньше многие люди в области компьютеров были идеалистическими «мечтателями», повёрнутыми на технологиях, но во время бума доткомов Computer

new PhDs can't wait to stop doing science to get into management or to become start-up gazillionaires.

That said, when I teach at Columbia, I have many more good students apply to my course than I could possible accept and I enjoy teaching.

I'm sad that fewer people read books and serious papers these days. Blogs and videos just don't give you the same depth of understanding.

L: What do you think about open-source software projects as a subject to study C++ programming practices? Is it worth it, taking into account that large projects have appeared before many modern C++ features, and therefore they have some legacy limitations and home-grown features used instead of standard ones?

BS: Many large projects started back in the dark ages and carry with them code and thought patterns that gets in the way of modern techniques and elegance. On the other hand, some "modern" projects are overly clever with uncritical use of newer language features, libraries, and design theories. I wish I had a long list of libraries and projects that I could recommend – there are many thousands to choose from – but I have never had time to review for educational purposes. I think that key to all teaching is knowledgeable teachers who can go through code pointing out strengths and weaknesses – every project has some of both. "Modernizing" a project can be a very useful exercise; rarely easy, of course, but with proper guidance/supervision it can be a very effective teaching tool. The C++ Core Guidelines can help. For a project not to be sterile, students need some insights in and some interest in the application domain.

L: Finally, are there any serious changes in the overall vision of the general-purpose languages (programming approaches, etc.) from your point of view, which are noticeable during the last 10 years? Would you like to mention any recent-born programming languages which are an interesting experiment from your point of view? Sorry for the last question, looks like it is included in most of your interviews, but it's really difficult to resist :)

BS: It is a hard question to avoid ☺. There are so many "new" languages! So many new ideas and variations of older ideas! It is probably fair to say that every new language offers something to their initial core constituency. The problem is that fragmenting the developer commu-

Science стали воспринимать как схему быстрого обогащения. В неё повалили люди, не имеющие глубокого интереса к технологиям. Затем, после схлопывания доткомов в 2002, компьютеры стали восприниматься как «провальная схема быстрого обогащения». Сейчас мы вернулись во времена бума, когда люди толпятся вокруг компьютеров ради построения хорошей карьеры или средств к существованию. В этом нет ничего неправильного – трудолюбивые умные студенты заслуживают хорошей карьеры – но мне немного не хватает того идеализма. Может быть в численном выражении идеалистов, заинтересованных в чистой науке и технологиях, стало больше, чем тогда, но их заглушает огромное количество людей в поисках безопасного причала. Я думаю, это грустно, когда свеже-испеченные обладатели учёной степени ждут не дождутся, чтобы бросить заниматься наукой и пойти в менеджмент или стать начинающими миллионерами.

Тем не менее, когда я преподаю в Колумбийском университете, у меня намного больше заявок от хороших студентов на мой курс, чем я способен принять, и я получаю удовольствие от преподавания.

Грустно, что сейчас меньше людей читают книги и серьёзные статьи. Блоги и видео просто не могут дать такой же глубины понимания.

L: А что вы думаете о проектах open-source как о предмете для изучения практики программирования на C++? Стоит ли овчинка выделки, учитывая, что большие проекты появились до многих современных особенностей C++, и поэтому в них есть некоторые унаследованные ограничения, а взамен некоторого стандартного функционала используются собственные внутренние разработки?

BS: Многие большие проекты были начаты в давние тёмные времена и тянут с собой код и образ мысли, которые встают на пути современных технологий и элегантности. С другой стороны, некоторые «современные» проекты перегружены заумью за счёт некритичного использования новых языковых особенностей, библиотек и теорий разработки. Хотел бы я иметь длинный список библиотек и проектов, которые я мог бы порекомендовать – выбирать можно из многих тысяч – но у меня никогда не было достаточно времени, чтобы сделать такой обзор для образовательных целей. Я думаю, ключ к любому обучению это знающие учителя, которые могли бы пройтись по коду, показывая его сильные и слабые места – у каждо-

nity into several dozen sub-communities, each with their own terminology, concepts, language, and tool chains is a recipe for chaos. This can happen within a single organization or even within a small group.

I like languages and new ideas, but I spend most of my time on two tasks: (1) how to build reliable, maintainable, well-performing applications and (2) how to improve C++ by solving practical problems and incorporating good ideas to help with that. That leaves me with little time for thorough and fair discussion of the dozens of new languages. Also, people who don't know me couldn't be sure that I would be fair.

Most ideas don't just suddenly appear in a language in a useful form. They "bubble" up in the various communities as ideas, articles, libraries, experiments before they find a form where they – in combination with other ideas, features, and techniques – become useful tools. Therefore, my answer to "do you like language X?" is usually along the lines of "I like the idea of X's feature/technique, but you can't just take an individual feature out of one language and directly graft it into another – features don't exist in isolation." That said, I look at functional-programming style pattern matching. In addition to what functional languages usually do, I also want to use it to make the visitor pattern redundant (`http://www.stroustrup.com/OpenPatternMatching.pdf`, `http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2019/p1371r1.pdf`). I look at the nice things you can do with reflection, but am horrified by the cost and the opportunities for errors in run-time reflection, so for C++ I/we look towards static refection (`http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2018/p0954r0.pdf`, `http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2019/p1733r0.pdf`).

I still see C++'s combination of direct manipulation of machine resources plus zero-overhead abstraction as a winning combination to problems with resource constraints and significant complexity. Almost all newer languages fail to address both well. Through LLVM, C++ has become part of the base of many (most?) newer language.

Questions and Russian translation by Dmitriy Kostiuk.

го проекта ведь есть и то и другое. «Осовременить» проект может быть очень полезным упражнением; конечно, это редко оказывается простой задачей, но под правильным руководством это может быть очень эффективным инструментом обучения. C++ Core Guidelines могут быть в этом полезны. А для того, чтобы проект не был стерильным, студенты должны иметь некоторое представление о предметной области и интерес к ней.

L: И наконец последний вопрос: есть ли с вашей точки зрения какие-то заметные изменения в общем взгляде на языки программирования общего назначения (подходы к программированию, и др.) – что-то за последние 10 лет? Может быть, вы хотели бы упомянуть какие-то недавно появившиеся языки программирования, которые на ваш взгяд стали интересным экспериментом? Извините за этот последний вопрос, он наверняка оказывается во всех ваших интервью, но от него действительно очень трудно удержаться :)

BS: Да, этого вопроса действительно трудно избежать ☺. «Новых» языков так много! И так много новых идей, или вариаций старых идей! Наверное, будет справедливо утверждать, что каждый новый язык предлагает что-то в своём первоначальном виде. Проблема в том, что фрагментация сообщества разработчиков на несколько дюжин меньших сообществ, каждое со своей собственной терминологией, концепциями, языком,и набором инструментов – это рецепт хаоса. Это может случиться и в рамках одной организации, и даже в рамках небольшой группы.

Мне нравятся языки и новые идеи, но большую часть своего времени я трачу на две задачи: (1) как строить надёжные, сопровождаемые приложения с хорошей производительностью, и (2) как улучшить C++ за счет решения практических задач и добавления хороших идей, которые могут в этом помочь. Из-за этого у меня остаётся мало времени на тщательное и беспристрастное обсуждение дюжин новых языков. А ещёи люди, которые меня не знают, не могут быть уверены, что я буду беспристрастен.

Большинство идей не появляются внезапно в языке в пригодном для употребления виде. Они сначала «всплывают» в различных сообществах в виде идей, статей, библиотек, экспериментов, и только потом находится та форма, в которой они – в сочетании с другими идеями, особенностями и технологиями – превра-

щаются в полезные инструменты. Поэтому мой ответ на вопрос «нравится ли вам язык X» обычно что-то вроде «Мне нравится идея такой-то особенности X, но нельзя взять какую-то особенность одного языка и просто перенести её в другой – они не могут существовать в изоляции». Тем не менее, я смотрю на сопоставление шаблонов в стиле функционального программирования. В дополнение к тому, что обычно делают функциональные языки программирования, я также хочу использовать это чтобы сделать ненужным паттерн «посетитель» (`http://www.stroustrup.com/OpenPatternMatching.pdf`, http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2019/p1371r1.pdf). Я смотрю на те хорошие вещи, которые можно делать с помощью рефлексии, но меня приводят в ужас затраты и возможности для ошибок в случае рефлексии в момент выполнения, поэтому для C++ я/мы нацелены на статическую рефлексию (http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2018/p0954r0.pdf, http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2019/p1733r0.pdf).

Я всё ещё вижу в реализованном в C++ сочетании прямого управления ресурсами компьютера плюс абстракции с нулевыми накладными расходами выигрышную комбинацию для задач с ограничениями по ресурсам и значительной сложностью. Почти все новые языки не могут хорошо справиться сразу с тем и другим. А через LLVM, C++ стал частью основы многих (а то и большинства?) новых языков.

Вопросы и русский перевод Дмитрия Костюка.