Руслан Богатырев

Летопись языков. Си

Источник: Мир ПК, #08/2001

Язык Си не имеет смысла представлять особо. За три десятилетия своей жизни он покорил сердца не одного миллиона программистов всего мира.

Си — это инструмент, острый, как бритва:
с его помощью можно создать
и элегантную программу, и кровавое месиво.
Брайан Керниган

На языке Си было создано такое количество программного обеспечения, с которым сравниться не может, пожалуй, ни один другой язык. История Си полна мифов и ложных стереотипов. Один из самых распространенных — авторство языка. Многие зачастую допускают одну и ту же ошибку, причисляя Брайана Кернигана к числу разработчиков Си. Он не участвовал в создании языка, но был активным его популяризатором и в соавторстве с Деннисом Ритчи написал бестселлер «Язык программирования Си» [1]. Впрочем, разве дело в мифах? Гораздо важнее понять, с чем связан небывалый успех Си, в чем секреты его долголетия и какое будущее ему уготовано? На эти вопросы мы и попробуем найти ответ, обратившись к страницам истории.

Истоки языка Си

Язык Си, как и любой другой, зародившийся в ходе эволюции программирования, появился не на пустом месте. У него были свои предшественники, изучая которые можно отыскать ключ к разгадке свойств и механизмов Си.

Сама атмосфера 1960-х годов способствовала возникновению чего-либо подобного. Питер Мойлан так пишет об этом [2]: «Нужен был язык, способный обойти некоторые жесткие правила, встроенные в большинство языков высокого уровня и обеспечивающие их надежность. Нужен был такой язык, который позволил бы делать то, что до него можно было реализовать только на ассемблере или на уровне машинного кода. Это привело к появлению концепции машинно-ориентированных языков промежуточного уровня. Надо признать, что Си был не единственным подобным языком и уж никак не самым первым. В сущности, в это время машинно-ориентированные языки стали появляться, как грибы после дождя. Все они походили один на другой, как члены единой семьи, но вовсе не потому, что авторы списывали их друг у друга, а потому, что они были под влиянием одних и тех же идей, витавших в то время в воздухе».

Главным прародителем языка Си стал BCPL. Его разработал Мартин Ричардс, когда в 1966–1967 гг. посещал Массачусетский технологический институт (MIT). Это была несколько упрощенная версия языка CPL (Cambridge Programming Language). Первоначально компилятор BCPL был реализован для операционных систем GECOS и MULTICS, работавших на компьютерах компании General Electric моделей GE-635 и GE-645 соответственно. BCPL использовался для реализации операционной системы TRIPOS, которая впоследствии была положена в основу AmigaDOS. Близким по духу к BCPL является язык BLISS (Basic Language for Implementation of System Software), созданный Биллом Вульфом в университете Карнеги-Меллон в 1969 г.

В 1969 г. Дуг Макилрой реализовал язык TMG (Макклар, 1965), придуманный специально для создания компиляторов. Кен Томпсон, вдохновленный работой Макилроя, решил, что для операционной системы UNIX (правда, тогда она еще так не называлась) нужен свой язык системного программирования. После нескольких неудачных экспериментов с Фортраном он создал такой язык и назвал его B (Би). По сути, B — это синтаксически видоизмененный BCPL, который Томпсону удалось втиснуть в 8 Кбайт памяти. Почему же он получил такое имя? Существует две гипотезы его происхождения: от начальной буквы либо BCPL, либо другого языка Томпсона — Bon, названного в честь его жены Бонни.

Многие знают о том, что компилятор языка Си был написан на самом Си (подобно тому, как несколько раньше группа Никлауса Вирта реализовала компилятор Паскаля [3] на Паскале). Однако гораздо менее известен тот факт, что его создатели во многом пошли по следам компилятора языка

B, который был написан Кеном Томпсоном на самом B. В результате работ Макилроя появился компилятор компиляторов TMGL. С его помощью была проведена программная раскрутка (bootstrapping) сначала языка B, а затем и Си. Иными словами, компиляторы для этих языков были написаны на тех же самых языках.

Таблица 1. Предшественники языка Си

Язык	Год	Автор
BCPL	1966	М.Ричардс
Bon	1968	К.Томпсон
B	1969	К.Томпсон
NB	1971	Д.Ритчи

Компилятор языка B впервые появился на компьютере PDP-7, где он генерировал не машинные инструкции, а шитый код — интерпретационную схему, где компилятор генерирует последовательность адресов, обозначающих фрагменты кода. Эти фрагменты и производят элементарные операции. В случае компилятора B операции выполнялись на простой стековой машине. Важным технологическим достижением группы, куда входили Томпсон и Ритчи, стал кросс-компилятор языка B: он был написан на самом B, работал на 18-разрядной PDP-7, генерировал код для 36-разрядной GE-635 и при этом умещался в 4 тыс. слов памяти PDP-7. Интересно, что у языка B первый компилятор занимал объем памяти 8 Кбайт, а у языка Си — 16 Кбайт.

Связанные родственными узами, языки BCPL, B и Си различаются синтаксисом, хотя и имеют общий фундамент. Они ориентированы на системное программирование. Написанные на них программы состоят из последовательности глобальных описаний и описания функций (процедур). Причем в BCPL процедуры могут быть вложенными, а в B и Си — нет. В отличие от своих потомков, BCPL имел очень широкий набор управляющих конструкций: **if-then, test-then-else, unless-do, while-do, until-do, repeat, repeatwhile, repeatuntil, for-to-by-do, loop, break, switchon-into-case-default-endcase**. Языки BCPL и B — бестиповые (т. е. не имеют типов), они работают со словом (ячейкой памяти), содержащим фиксированное число разрядов (битов), а память ими рассматривается как линейный массив слов, где значение ячейки памяти можно интерпретировать как индекс в этом массиве. Для всех этих целей BCPL использует оператор «!», а язык B — оператор «*».

Деннис Ритчи вспоминает [4]: «В 1971 г. я начал расширять язык B, добавляя тип char, а также переписал его компилятор таким образом, чтобы он впрямую генерировал инструкции для PDP-11, а не шитый код. Таким образом, переход от B к Си происходил одновременно с созданием компилятора, способного порождать достаточно быстрые и компактные программы в сравнении с языком ассемблера. Я назвал несколько расширенный язык NB — «новый B» (new B). В язык NB Ритчи ввел первые типы: int и char. Вместе с массивами и указателями они составили его систему типов.

Откуда же появилось название Си? Ритчи разъясняет это так: «Создав систему типов, соответствующий синтаксис и компилятор для нового языка, я почувствовал, что он заслуживает нового имени: NB показалось мне недостаточно четким. Я решил следовать однобуквенному стилю и назвал его C (Си), оставляя открытым вопрос, являлось ли после B это следующей буквой в алфавите или в названии BCPL».

Си, UNIX и PDP-11

Успех Си был неразрывно связан с тем, что в одном месте в одно и то же время появились сразу три грандиозных творения, ставших культовыми: язык программирования Си, операционная система UNIX и мини-компьютер PDP-11. (В Советском Союзе аналогом PDP-11 были семейства СМ-4 и СМ-1420.) За прошедшие десятилетия PDP-11 и сменивший его VAX-11 уступили место более совершенным техническим решениям, но Си и UNIX продолжают оставаться на передовых позициях программной индустрии. Они связаны друг с другом столь тесно, что невольно хочется перефразировать слова Маяковского: мы говорим «Си», подразумеваем «UNIX», мы говорим «UNIX», подразумеваем «Си».

Связка Си—UNIX во многом отразила творческий союз Ритчи и Томпсона. Кен Томпсон считает: «Наше сотрудничество было образцом совершенства. За те десять лет, что мы проработали вместе, можно вспомнить только один случай нескоординированной работы. Я тогда обнаружил, что мы написали одинаковую ассемблерную программу из 20 строк. Я сравнил наши тексты и поразился, обнаружив, что они совпадают посимвольно. Результат нашей совместной работы получился намного более значительным, чем вклад нас обоих по отдельности».

В создании UNIX участвовали шесть человек. Ритчи вспоминает [5]: «UNIX была разработана Кеном Томпсоном. Я написал много системного ПО, Кен — большую часть остального. Среди других участников были Джо Оссанна, Дуг Макилрой и Боб Моррис». Шестым, о ком забыл упомянуть Ритчи, был Ричард Кенедей.

Таблица 2. Этапы создания канонического варианта ОС UNIX

Год	Версия	Комментарии
1969	V1	На ассемблере на PDP-7 и PDP-9
1971	V2	PDP-11/20 без защиты памяти
1972	V3	PDP-11/34, /40, /45, /60, /70, каналы, мультипрограммирование
1973	V4	Работала на PDP-11/70 и Interdata 8/32, ядро переписано на Си
1975	V6	Первая версия, доступная за пределами Bell Labs
1979	V7	"Истинная UNIX", ядро занимало 40 Кбайт

Катализатором появления UNIX стала система MULTICS (Multiplexed Information and Computing Service), которая, как и UNIX, используется по сей день. Она возникла как результат проекта MAC (Multiple Access Computers), стартовавшего в ноябре 1962 г. в MIT. В конце 1964 г. к MIT присоединились компания General Electric и лаборатории AT&T Bell Labs. Неудовлетворенность руководства Bell Labs темпами работ привела его к решению выйти из проекта. Разрыв состоялся в апреле 1969 г., и небольшой коллектив, куда помимо Томпсона и Ритчи входили Джо Оссанна, Дуг Макилрой, Боб Моррис и Ричард Кенедей, оказался не у дел. Все их попытки убедить руководство в необходимости приобрести подходящий полигон для реализации идей по созданию новой ОС (речь тогда шла о мэйнфреймах DEC PDP-10 и SDS Sigma 7) ни к чему не привели. Вооружившись мелом и пером, Томпсон, Ритчи и Кенедей начали проектировать новую иерархическую файловую систему. Она стала, по словам Ритчи, сердцем UNIX, а все остальное строилось именно на ее основе.

Как вспоминает Деннис Ритчи [6], название UNIX предложил Брайан Керниган. Оно перекликалось с названием системы MULTICS. Причем существуют две версии образования имени UNIX: по одной — это просто трансформация названия Multics («multi» заменяется на «uni», а «cs» — на «x»), по другой — видоизмененная аббревиатура UNiplexed Information and Computing System (UNICS). Если вспомнить, что MULTICS само по себе сокращение, то обе версии сливаются в одну.

Деннис Ритчи вспоминает [7]: «Операционная система UNIX внезапно стала новинкой, хотя таковой и не была. Она зародилась в 1969 г., когда Кен Томпсон открыл для себя малоиспользуемую машину PDP-7 и решил оснастить ее программной средой на свой вкус. Его работа скоро привлекла меня, и я присоединился к нему. Следует уточнить: многие идеи и основная часть работы по их воплощению принадлежали ему».

Несколько слов о той технике, на которой все это создавалось. В те времена первый компьютер фирмы Digital (DEC), PDP-1, стоил дешевле, чем канал ввода-вывода, применявшийся на лучшей тогда модели IBM 7090. Он представлял собой 18-разрядную машину, что сейчас некоторым может показаться странным. Преемницей PDP-1 (1960) стала PDP-4 (1962), а на смену ей пришла PDP-7 (1965). К началу 1970-х годов было продано 120 таких машин. Система ввода-вывода компьютера PDP-7 обладала довольно интересной особенностью: в то время промышленность переходила на стандарт ASCII, поэтому PDP-7 спроектировали так, чтобы можно было принимать как ASCII, так и 5-разрядный код Бодо, ранее активно применявшийся на телетайпе.

Работа корпорации Digital над 16-разрядным семейством PDP-11 стала поворотным пунктом в ее отношении к программному обеспечению. Так, в проекте PDP-11 появилось сразу несколько программных нововведений. Среди них — время срабатывания прерывания, вспомогательные средства отладки, генерируемый компилятором код, загрузка программ и

распределение памяти. Как и в случае с семейством 18-разрядных машин, главным языком PDP-11 стал Фортран. Популярность PDP-11 оказалась огромной: за первые восемь лет производства (с 1970 г.) было продано 50 тыс. компьютеров.

В 1971 г. Ритчи и Томпсон получили в свое распоряжение машину PDP-11 и к концу года обеспечивали работу трех первых пользователей — машинисток, вводящих заявки на патенты в Мюррей-Хилл, где размещались лаборатории Bell Labs. Перенос UNIX с PDP-7 на PDP-11 произошел в первом квартале 1971 г. На язык Си ОС UNIX была переписана в 1973 г. и тогда же представлена широкой общественности на Конференции по принципам операционных систем. Первой важной публикацией стала статья Ритчи и Томпсона в журнале «Communications of the ACM» [8]. В состав ПО в UNIX входили штатные для PDP-11 ассемблер и компилятор Фортрана, компилятор языка B (Кен Томпсон), а также созданные Дугом Макилроем макропроцессор M6 и компилятор компиляторов TMGL — предшественник компилятора YACC Стива Джонсона. Вслед за реализацией на PDP-11 в начале 1970-х годов появились компиляторы Си для компьютеров IBM 360/370, Honeywell 635, Interdata 8/32.

Становление и развитие языка

Эволюцию Си можно условно разбить на три стадии: детство (1971–1977), юность (1978–1988) и зрелость (с 1989 г.). В табл. 3 показаны основные вехи «большого пути».

Таблица 3. Эволюция языка Си

Год	Комментарии
1971	Язык NB
1972	Новая система типов, переименование NB в C, первая версия языка Си
1972	Введение препроцессора (директивы #include и #define)
1973	Язык подготовлен для написания ядра UNIX
1977	Надежность типов и переносимость, добавление unsigned, long, union, struct
1978	K&R — канонический вариант Си, изложенный в книге Кернигана-Ритчи
1983	По инициативе Макилроя для Си создан комитет по стандартизации (ANSI X3J11)
1988	K&R 2 — второе издание книги Кернигана-Ритчи
1989	Принятие техническим комитетом X3J11 американского стандарта ANSI X3.159-1989
1990	Международный стандарт ISO/IEC 9899-1990
1999	Стандарт C99 (ISO/IEC 9899-1999)

На первой стадии основное внимание уделялось технической стороне дела, на второй — во главу угла были поставлены популяризация и формирование сообщества пользователей, на третьей (после стандартизации) — начался процесс активного промышленного использования языка.

Одним из определяющих событий стало появление в 1978 г. бестселлера K&R, иначе называемого «белой книгой» [1]. Он де-факто установил стандарт языка и сделал его каноническим как минимум на целое десятилетие. Ритчи вспоминает: «Керниган написал почти весь пояснительный материал, я же отвечал за приложение, куда было включено справочное руководство по языку, и за главу по взаимодействию с UNIX». Вот так и раскрылся секрет: язык создал Ритчи, а книгу в основном написал Керниган.

Де-юре язык Си был стандартизирован в 1989 г. Американским национальным институтом стандартов (ANSI). Практически тот же самый стандарт был принят Международной организацией по стандартизации (ISO) годом позже. Формально с появлением ISO-стандарта ANSI-стандарт был изъят, но вместо ISO в среде разработчиков было принято все же говорить о диалекте ANSI C.

По мнению Брайана Кернигана, «главные достоинства Си состоят в том, что он предоставляет программисту возможность полного управления реализацией и что программы, написанные на Си, работают, как правило, весьма быстро». Большое влияние на Си, помимо BCPL и B, оказал Алгол-68, в частности это касается схемы композиции типов, конструкций struct и union,

приведения типов. Важным изменениям язык подвергся в 1972–1973 гг., когда шла подготовка к «погружению» в него ОС UNIX. Самым Алана значительным из них стал препроцессор, на создание которого наибольшее влияние произвели идеи Шнайдера и механизмы импорта файлов в языках BCPL и ПЛ/1. Условная компиляция и макросы в Си появились в тот же период благодаря работам Майка Леска и Джона Рейзера. Что касается стандартной библиотеки ввода-вывода, ставшей важной составляющей популярности и практической ценности языка, то она была разработана в 1973 г. там же, в Bell Labs, Майком Леском.

«Си — это инструмент, острый, как бритва: с его помощью можно создать и элегантную программу, и кровавое месиво». В этом малоизвестном утверждении Кернигана скрыт важный смысл. Он пытается отвратить программистов от бездумного применения всех имеющихся в их распоряжении средств. «Какова роль языка? — размышляет он. — Основной движущей силой эволюции языков программирования стала попытка предотвратить ошибки, используя возможности этих языков. Одни из таких возможностей уменьшают шанс появления целых классов ошибок: проверка диапазонов индексов, ограничение использования указателей или полный отказ от них, сборка мусора, строковые типы данных, типизированный ввод-вывод, строгая проверка типов. Однако другие возможности языка напрашиваются на ошибку, например оператор goto, глобальные переменные, свободно применяемые указатели, автоматические преобразования типов. Программистам следует знать зоны повышенного риска в своих языках и обращаться с ними особенно осторожно».

Сегодня Си, как и Паскаль, нередко подвергают критике, во многом вполне справедливо. Даже сам Керниган признает: «Сравнивая способы, которыми программы на Си и Java представляют и обрабатывают одни и те же структуры данных, следует отметить, что в случае Java функциональные обязанности разделены лучше». Бывают и более резкие оценки. Так, Питер Мойлан не без сарказма замечает: «Бытует мнение, что Си апеллирует к «мужскому началу» программистов, которым нравится сражаться с малопонятными ошибками и находить невероятные и хитроумные решения проблем. Многих привлекает и компактность нотации Си. Похоже, сторонники этого языка считают, что возможность написать такой, скажем, оператор, как **p++^=q++=*r-s, служит серьезным аргументом в пользу применения Си, поскольку экономится время. Скептик может возразить: такое ускорение сведет на нет необходимость в дополнительных комментариях. Достаточно просмотреть несколько типичных программ на языке Си, чтобы убедиться: на комментариях здесь тоже экономят, причем даже так называемые профессиональные программисты».

Ахиллесовой пятой языка Си, как и в случае классического Паскаля, является отсутствие поддержки модулей (хотя бы в понимании Ады и Модулы-2) и, следовательно, полноценной раздельной компиляции. Суррогат в виде препроцессорных директив «#include» не спасает ситуацию, а лишь усугубляет ее еще больше. Вместо раздельной компиляции получается независимая, при которой вопросы рассинхронизации интерфейса и реализации решаются весьма замысловатым и ненадежным образом. Что уж тут говорить о поддержке контрактных форм интерфейсов, исповедуемых Бертраном Мейером, автором языка Eiffel. Но Кернигану надо отдать должное. Критику он признает и отмечает: «Препроцессор Си — мощный, но несколько туповатый инструмент, а макросы — вообще довольно опасная вещь, поскольку они изменяют лексическую структуру программы».

И все же недостатки — оборотная сторона достоинств. Если бы в отношении Си это было бы не так, вряд ли язык дал бы жизнь многим своим потомкам (см. табл. 4), среди которых самым известным стал C++. При создании C++ Бьерн Страуструп искал базовый язык для расширения его концепцией классов из языка Симула. Как альтернатива Си, по словам Страуструпа, рассматривались Модула-2, Ада, Smalltalk, Mesa и CLU. Но предпочтение было отдано все же Си.

Таблица 4. Прямые потомки языка Си

Язык	Год	Автор
C++	1984	Б. Страуструп
Concurrent C	1986	Н. Джехани
Objective C	1986	Б. Кокс
C*	1987	Thinking Machines
C+@, ранее Calico	1991	Bell Labs

Будущее языка Си

Появление благодаря работам Бьерна Страуструпа языка C++ (1984), ставшего фактически главным языком программной индустрии, а также повальное увлечение объектно-ориентированным проектированием и программированием заставили уйти в тень его именитого предшественника — язык Си. С другой стороны, в ходе программной эволюции возникли два направления, которые закрепили за языком Си статус безусловного лидерства: API-программирование и перенацеливаемые компиляторы. Программные интерфейсы (API), порождаемые бесчисленными производителями ПО и понимаемые в широком смысле как интерфейсы системного и прикладного программирования, формируются с прицелом на Си. Остальные языки вынуждены либо пользоваться этим слоем, либо подстраивать его под себя, что в условиях безудержной гонки версий создает массу проблем. Итак, при формировании программной инфраструктуры Си стал основным связующим слоем между разнородными системами и компонентами.

Другой его конек — генерация промежуточного кода с настройкой на особенности целевой платформы. Если раньше при разработке компиляторов кодогенератор «затачивался» под конкретную аппаратную и операционную платформу, то сейчас требуется гибкость перенацеливания генерации. А ее проще всего бывает обеспечить, если выбрать Си в качестве языка промежуточного представления программного кода, сводя, как в математике, новую задачу к уже решенной. Брайан Керниган особо отмечает эту важную роль языка Ритчи: «Си часто используется в качестве языка ассемблера высокого уровня. Modula-3 и C++ относятся к тем языкам общего назначения, для которых первые компиляторы создавали код на Си, обрабатывающийся уже затем стандартным компилятором... Подобный подход очень помог этим языкам на ранних стадиях их внедрения».

Неизменность языка Си на протяжении многих лет и стала одной из серьезных предпосылок для его лидерства в упомянутых областях. Теперь же, с выходом стандарта C99, ситуация становится менее ясной. Си фактически превращается из подмножества C++ в самостоятельно развивающийся язык, нарушая сложившуюся в последние годы и уже успевшую укорениться структуру связей, когда эти языки применялись неразрывно.

Бьерн Страуструп высказывает серьезную озабоченность подобными обстоятельствами: «Новшества Си-99 (C99) касаются расширения низкоуровневых средств Си в области численного программирования, а включенные в C++ средства абстрагирования и универсализации в основном проигнорированы. Это усложняет достижение совместимости, поскольку в Си добавляются специализированные возможности для конкретных случаев, а в C++ те же потребности программиста удовлетворяются с помощью библиотек, реализованных с применением языковых средств общего назначения».

По поводу нового стандарта и сам Ритчи выражает свое мнение сдержанно-скептически [9]: «Я был удовлетворен стандартом ANSI/ISO 1989–1990 гг. Новый стандарт C99 намного более громоздкий, и хотя комитет сообщил, что потратил значительную часть своего времени на отсеивание предложений о нововведениях, принял он тоже немало, и все это еще предстоит переварить. Я определенно не хочу дополнительных возможностей и, очевидно, предпочел бы, чтобы комитет сопротивлялся более стойко... Стандарт C99 мне не очень нравится, однако я не пытаюсь его отменить. Комитет отлично поработал; Си действительно должен развиваться».

Однако где проходит та грань, которую при внесении изменений в язык нельзя переступать? Главное, чтобы новации не приводили к еще более серьезным проблемам, к появлению таких побочных эффектов, которые перечеркнули бы весь выигрыш от новых чудодейственных «препаратов». Тони Хоар в этой связи отмечает: «Программистам всегда приходится соприкасаться со сложностью, и этого нельзя избежать. Наши приложения сложны, потому что мы честолюбивы и стремимся использовать наши компьютеры все более сложными способами. Программирование сложно из-за большого числа противоречивых целей, преследуемых каждым программным проектом. Если наш основной инструмент, язык, на котором мы составляем и кодируем программу, также непрост, то сам язык становится частью нашей проблемы, а не ее решения. Когда какой-нибудь проект нового языка близится к завершению, то всегда возникает безумная спешка — нужно внести новые свойства еще до стандартизации. Это стремление действительно безумно, потому что оно заводит в ловушку, из которой нет выхода. Недостающие свойства всегда можно добавить позже, когда их конструкция и последствия будут хорошо поняты. Свойство, включенное до того, как стало понято, никогда нельзя изъять позже».

С такой позицией солидарен и другой классик программирования — Эдсгер Дейкстра: «Я вспоминаю одну лекцию на симпозиуме по языкам программирования высокого уровня, прочитанную в защиту языка ПЛ/1 человеком, который называл себя его преданным пользователем. Но в конце этой

часовой лекции, восхваляющей ПЛ/1, он ухитрился попросить добавления около 50 новых «свойств», совсем не предполагая, что основным источником его трудностей является именно то, что язык и так имеет слишком много «свойств». Лектор проявил все удручающие симптомы наркомании: дошедший до отупения, он мог лишь просить все больше, больше, больше...»

Бьерн Страуструп все же надеется, что разрыва между Си и C++ удастся избежать. Он подчеркивает: «Моим идеалом по-прежнему остается единый язык, и техническая возможность слить C++ с C99 пока сохраняется. На мой взгляд, такой язык мог бы соответствовать любым разумным техническим условиям. Однако я не уверен, что это приемлемо политически. Для начала потребовалось бы объединить комитеты по стандартизации Си и C++: нельзя было бы иметь две различные группы, параллельно развивающие два языка».

Вне зависимости от того, по какому пути пойдет Си, возникает вопрос: останется ли Си и через пять или десять лет таким же популярным и незаменимым, как сейчас? Деннис Ритчи на сей счет высказывается очень осторожно: «Я, честно говоря, не знаю ответа на этот вопрос. Могу только заметить, что программное обеспечение в целом труднее поддается замене, чем аппаратное. Скажем, число программ на C++ и Java, вероятно, растет быстрее, чем число программ на обычном Си, но я готов спорить, что Си сохранится. Инфраструктура не позволит его вытеснить. То же самое, конечно, можно сказать и о других языках (например, о версиях Паскаля, о языке Ада)».

В чем же секрет феномена Си? Деннис Ритчи в своем выступлении на Международной конференции по истории языков программирования (1993) сформулировал разгадку так: «Язык Си — это невероятный, оглушительный и огромный успех. Хотя повороты судьбы ему помогали, он безусловно удовлетворил потребности в таком языке реализации систем, который был достаточно эффективным, чтобы заменить ассемблер, и в то же время достаточно абстрактным и гибким, чтобы описывать алгоритмы и взаимодействия в широком спектре программных сред». Да, Си, несмотря на справедливую критику в его адрес и техническое несовершенство, все же сумел завоевать мир. Ибо — такова жизнь. «Существует масса прекрасных языков (прекраснее, чем Си), которые не привились,— признает Ритчи,— но кто-то все же выигрывает...»

Литература

1. Kernighan B.W., Ritchie D.M. The C Programming Language. Prentice-Hall, 1978. [Пер. на рус. яз.: Керниган Б., Ритчи Д. Язык программирования Си. М.: Финансы и статистика. 1985].
2. Moylan Peter. The case against C // The ModulaTor. — 1993. — V.3.—N 6.
3. Богатырев Р. Летопись языков. Паскаль // Мир ПК. — 2001. — № 4.
4. Ritchie D.M. The Development of the C Programming Language // History of Programming Languages. ACM-Press, 1996.
5. Ritchie D.M. Writings from the Past. 1997.
6. Ritchie D.M. The evolution of the UNIX time-sharing system // Language Design and Programming Methodology/Ed. by J.Tobbias, Springer, 1980.
7. Ритчи Д. Размышления об исследованиях в области программного обеспечения // Лекции лауреатов премии Тьюринга за первые двадцать лет. 1966–1985. — М.: Мир, 1993.
8. Ritchie D. M., Thompson K. The UNIX Time-Sharing System // Communications of the ACM. — 1974.— V.17. — N7.
9. Калев Д. Будущее по Дэннису Ритчи // Мир ПК. — 2001. — №3.
10. Мосени П. Inferno: ОС для сетевых компьютеров // Мир ПК. — 1997. — №10.

Биография

Деннис Ритчи (Dennis M. Ritchie) родился 9 сентября 1941 г. в Бронксвилле (шт. Нью-Йорк). Закончил Гарвардский университет, имеет степень бакалавра по физике и магистра по прикладной математике. Темой диссертации, защищенной в 1968 г., была субрекурсивная иерархия функций. В 1967 г., продолжая семейные традиции и пойдя по стопам отца, поступил на работу в исследовательскую лабораторию AT&T Bell Laboratories в Мюррей-Хилле (шт. Нью-Джерси). Ритчи участвовал в разработке системы MULTICS (совместный проект Bell Labs, MIT и General Electric), а также компилятора языка BCPL на компьютерах GE 645 (ОС MULTICS) и GE 635 (ОС GECOS). В этот же период он написал компилятор для языка ALTRAN, предназначенного для ведения символьных вычислений. Активно помогал Кену Томпсону в создании знаменитой операционной системы UNIX. В самом начале разработки UNIX добавил систему типов и новый синтаксис к языку B («Би»), созданному Томпсоном. Так родился Си. Совместно со Стивом Джонсоном и Кеном Томпсоном участвовал в переносе UNIX на Interdata 8/32. ОС UNIX 7 Edition, вышедшая из отдела Томпсона и Ритчи, стала основой для UNIX BSD и коммерческой UNIX System V. После реорганизации AT&T, уже в рамках Lucent Labs, возглавлял группу по созданию операционной системы Plan 9 (1995). Она была положена в основу следующего проекта группы Ритчи — ОС Inferno, анонсированной в апреле 1996 г. Для этой системы Ритчи разработал язык Limbo [10]. В настоящее время он возглавляет отдел исследований по системному ПО в Bell Laboratories. В 1983 г. на ежегодной конференции ассоциации ACM Деннису Ритчи и Кену Томпсону за разработку и реализацию языка программирования Си и операционной системы UNIX была вручена престижная премия Тьюринга. Отмечен многими наградами, среди которых премия Эммануэля Пьора (1982), премия ACM Software Systems Award (1983) и медаль Хэмминга (1990). В 1988 г. Ритчи избран в Американскую национальную инженерную академию (National Academy of Engineering).