Содержание

Предисловие

Это ли истинный панк-рок,
Верный, как линия партии?
Wilco, «Too Far Apart»

Язык C и панк-рок

В языке C совсем немного **ключевых** слов, немало острых углов и безграничные возможности. Он позволяет сделать абсолютно все. Его изучение можно сравнить с гитарными аккордами C, G и D – основные движения освоить легко, а потом всю жизнь можно совершенствоваться. Отвергающие C боятся скрытой в нем мощи, считая ее небезопасной. По всем рейтингам C неизменно занимает первое место среди языков, продвижение которых не спонсируется никакими корпорациями или фондами[1].

Языку уже 40 лет, то есть он достиг среднего возраста. Ребята, создавшие его, сделали это вопреки желанию руководства – полная аналогия с истоками панк-рока, – но случилось это в 1970-х годах, и у языка было достаточно времени, чтобы стать популярным.

Что происходило, когда панк-рок стал популярным? Зародившись в 1970-х годах, панк, безусловно, вышел с обочины на большую дорогу. Тиражи альбомов таких групп, как The Clash, The Offspring, Green Day и The Strokes, исчисляются миллионами экземпляров, а в местном супермаркете мне доводилось слышать легкие инструментальные переложения песен в стиле отпочковавшегося от панк-рока музыкального жанра «грандж». Бывший солист группы «Слиттер-Кинни» теперь ведет популярное комедийное шоу, в котором часто язвительно пародируются панк-рокеры[2]. В ответ на продолжающуюся эволюцию можно было бы занять жесткую позицию и сказать, что панк – это только то, что было в начале, а все остальное – легонький поп-панк для массовой аудитории. Блюстители традиций могут слушать свои пластинки 70-х годов, а когда бороздки износятся – скачать оцифрованное издание. А своих малолетних отпрысков одеть в «кенгурушки» – ностальгируя по группе «Рамонес».

Чужим этого не понять. Кто-то, слыша слово «панк», представляет себе канувшее в историю явление 1970-х годов – что-то связанное с парнями, которые делали нечто необычное. Традиционалисты, которые все еще любят и слушают диски Игги Попа, ловят свой кайф, но от того впечатление, что панк закостенел и уже неактуален, лишь усиливается.

---

[1] Это предисловие, вне всяких сомнений, многим обязано статье Криса Адамсона «Punk Rock Languages: A Polemic» по адресу http://pragprog.com/magazines/2011-03/punk-rock-languages.

[2] С такими стихами, как «Can't get to heaven with a three-chord song», быть может, Слитера-Кинни стоило отнести к постпанку? К сожалению, на панк нет стандарта ИСО, на который можно было бы ориентироваться, решая, кого куда отнести.

Однако вернемся в мир C. Тут тоже есть как традиционалисты, размахивающие знаменем со словами ANSI 89, так и люди, готовые использовать все, что реально работает, и даже не знающие, что код, который они пишут, в 1990-е годы нельзя было бы откомпилировать или запустить. Чужаки не замечают разницы. Они видят написанные в 1980-е книги, которые все еще лежат на прилавках, читают написанные в 1990-е онлайновые пособия, внимают упертым традиционалистам, которые настаивают, что и сегодня нужно писать, как тогда, и даже не понимают, что сам язык и его пользователи не застыли в развитии. Печально это – ведь они отказываются от поистине замечательных вещей.

Эта книга о том, как порвать с традицией и вернуть C новизну панк-рока. Я не собираюсь сравнивать свой код с оригинальной спецификацией, изложенной в книге Кернигана и Ричи 1978 года. В моем смартфоне 512 мегабайт памяти, так зачем же авторы учебников по C продолжают на десятках страниц наставлять, как сократить размер исполняемого файла на несколько килобайтов? Я пишу этот текст на дешевеньком нетбуке, способном выполнять 3 200 000 000 машинных команд в секунду, так какое мне дело до разрядности операндов команды: 8 или 16? Мы же в любом случае пишем на C, поэтому наш удобочитаемый, но неидеально оптимизированный код все равно будет работать на порядок быстрее, чем сравнимый код на любом другом распухшем от обилия функциональности языке.

Вопросы и ответы (или о параметрах этой книги)

В. Чем эта книга отличается от других книг по C?

О. Одни книги лучше написаны, другие даже занимательны, но у всех учебников по C есть одна общая особенность (а я прочитал их *множество*, в том числе [Deitel 2013], [Griffiths 2012], [Kernighan 1978], [Kernighan 1988], [Kochan 2004], [Oualline 1997], [Perry 1994], [Prata 2004] и [Ullman 2004]). По большей части, они были написаны до выхода стандарта C99, в котором упрощены многие аспекты использования языка. А бывает и так, что в очередное издание книги просто включено несколько замечаний о новшествах, но нет никакого серьезного переосмысления способов работы с языком. Во всех говорится, что, возможно, существуют библиотеки, которые могут пригодиться в собственном коде, но, как правило, ни слова об инструментах установки и экосистеме, благодаря которой эти библиотеки оказываются надежными и в разумной степени переносимыми. Материал, изложенный в этих учебниках, по-прежнему остается в силе и не утратил ценности, только вот современный код на C мало напоминает тот, который приводится в предлагаемых примерах.

Эта книга начинается там, где другие заканчиваются. Сам язык и окружающая его экосистема подвергаются пересмотру. Основная сюжетная линия – как пользоваться библиотеками для работы со связанными списками и анализаторами XML, а не разрабатывать собственные с нуля. Это книга о том, как писать удобочитаемый код с дружественным пользователю программным интерфейсом.

В. На кого рассчитана эта книга? На экспертов по кодированию?

О. Предполагается, что у вас есть опыт кодирования на каком-нибудь языке, к примеру на Java или скриптовом языке типа Perl. Я не собираюсь объяснять, почему программа не должна быть одной длинной процедурой, не разбитой на функции.

В тексте книги предполагается, что читатель обладает базовыми знаниями о C, приобретенными в процессе написания кода на этом языке. Для тех, кто подзабыл детали или вообще начинает с азов, в приложении A приводится краткий справочник по основам C, рассчитанный на владеющих такими скриптовыми языками, как Python или Ruby.

Наверное, стоит упомянуть, что я написал также учебник по статистическим и научным расчетам «Modeling with Data» [Klemens 2008]. Помимо многочисленных деталей, относящихся к численным методам и использованию статистических моделей для описания данных, там имеется более развернутое и независимое руководство по C, в котором, надеюсь, мне удалось преодолеть многие недостатки прежних руководств.

В. Я прикладной программист и не собираюсь копаться в ядре. Зачем мне писать на C, а не на скриптовом языке Python, на котором программировать куда быстрее?

О. Если вы прикладной программист, то эта книга как раз для вас. Сколько раз я слышал утверждение, будто C – язык системного программирования; оно страшно далеко от панковского склада ума – да кто они такие, чтобы указывать нам, что можно писать, а что – нет?

Высказывания типа «наш язык почти такой же быстрый, как C, но писать на нем проще» уже набили оскомину. Понятно же, что сам C такой же быстрый, как C, а задача этой книги – убедить вас в том, что писать на нем проще, чем это следует из книг десятилетней давности. Вызывать `malloc` и забираться в дебри управления памятью вам придется даже вполовину не так часто, как системному программисту 1990-х годов. У нас теперь есть простые средства для работы со строками, и даже базовый синтаксис изменился, чтобы сделать код понятнее.

Я всерьез начал писать на C, когда понадобилось ускорить программу моделирования, написанную на скриптовом языке R. Как и многие другие скриптовые языки, R имеет интерфейс к C, которым предлагается пользоваться всякий раз, как включающий язык оказывается слишком медленным. В конечном итоге я переписал на C так много функций, что от включающего языка вообще отказался.

В. То, что эта книга понравится прикладным программистам с опытом работы на скриптовых языках, конечно, прекрасно, но я-то занимаюсь кодом ядра. Я выучил C в пятом классе, у меня даже сны иногда успешно компилируются. Что тут для меня может быть нового?

О. Последние 20 лет C не стоял на месте. Ниже я расскажу о том, как изменилась функциональность, гарантированно поддерживаемая любым компилятором, – благодаря двум новым стандартам C, вышедшим со времен оригинального стандарта ANSI. Загляните в главу 10, может статься, кое-что в ней вас удивит.

В некоторых частях книги, например в главе 6, развенчивающей широко распространенные ошибочные представления об указателях, рассматриваются вещи, изменившиеся с 1980-х годов.

Прогресс затронул и окружение. Многие рассматриваемые мной инструменты, например `make` и отладчик, вам, наверное, знакомы, но есть другие, не столь хорошо известные. Комплект инструментов Autotools полностью изменил представление о распространении кода, а система управления версиями Git знаменует новый подход к коллективной разработке кода.

В. Не могу не отметить, что примерно в трети книги вообще нет кода на C.

О. Цель этой книги – рассмотреть то, чего нет в других учебниках C, а первым номером в этом списке стоят инструменты и окружение. Если вы не пользуетесь отладчиком (автономным или входящим в IDE), то заметно усложняете себе жизнь. Во многих учебниках отладчик вынесен куда-то на задворки, если вообще упоминается. Для совместной работы над кодом нужен другой комплект инструментов, включающий среди прочего Autotools и Git. Код существует не в вакууме, и я полагал, что окажу читателям дурную услугу, написав еще одну книгу, основанную на предпосылке, будто знание синтаксиса – это все, что необходимо для продуктивного использования языка.

В. Есть много средств для разработки программ на C. Какими критериями вы руководствовались при отборе?

О. Сообщество пользователей C в большей степени, чем многие другие, озабочено следованием стандартам интероперабельности. Существует масса расширений C, предлагаемых в среде GNU, есть интегрированные среды (IDE), которые работают только в Windows, а также расширения компилятора, доступные лишь в LLVM (Low Level Virtual Machine – низкоуровневая виртуальная машина). Быть может, именно поэтому авторы прежних учебников боялись затрагивать тему инструментальных средств. Но в наши дни существуют системы, которые работают на всем, что мы обычно считаем компьютером. Многие являются частью проекта GNU; LLVM со своим инструментарием быстро набирают популярность, но пока еще не являются преобладающими. Где бы вы ни работали – в Windows, в Linux, на экземпляре, только что полученном от поставщика облачных вычислений, – рассматриваемые здесь инструменты можно будет установить легко и быстро. Я упомяну о нескольких платформенно-зависимых инструментах, но всякий раз буду явно отмечать это.

Я не рассматриваю интегрированных сред разработки (IDE), потому что вряд ли найдутся такие, которые надежно работают на любой платформе (попробуйте поставить Eclipse и подключаемые к ней модули для C на экземпляр Amazon Elastic Compute Cloud), да к тому же выбор IDE в высшей степени субъективен. В состав типичной IDE входит система сборки проектов, которая обычно несовместима с аналогичной системой сборки из другой IDE. Поэтому файлы проектов IDE невозможно использовать для распространения проекта; исключением являются случаи, когда все заинтересованные лица обязаны работать с одной и той же IDE (учебные курсы, некоторые офисы, некоторые вычислительные платформы).

В. У меня есть Интернет. Чтобы посмотреть справку по команде или нюансы синтаксиса, хватит пары секунд, так зачем мне читать книгу?

О. Истинная правда: чтобы посмотреть таблицу приоритетов операторов в системе Linux или Mac, достаточно набрать команду `man operator`. Почему же тогда я привожу ее в книге?

У меня точно такой же Интернет, как у вас, и я немало времени провожу в нем, читая разные материалы. Поэтому я прекрасно знаю, о чем там не пишут, и это именно то, что я включил в книгу. Описывая новый инструмент, например gprof или GDB, я сообщаю то, что необходимо знать, чтобы сориентироваться и задать поисковой системе разумные вопросы, а также то, о чем умалчивают другие учебники (а это ой как много).

Стандарты: как много девушек хороших

Если явно не оговорено противное, весь код в книге соответствует стандартам ISO C99 и C11. Чтобы вы понимали, о чем идет речь, будет уместно дать краткий исторический обзор и перечислить основные стандарты языка C (опуская мелкие редакционные правки и исправления).

K & R (примерно 1978)

Деннис Ричи, Кен Томпсон и ряд сподвижников придумали язык для написания операционной системы Unix. Впоследствии Брайан Керниган и Деннис Ричи привели описание языка в первом издании своей книги. Это и был первый стандарт *де-факто* [Kernighan 1978].

ANSI C89

Компания Bell Labs передала курирование языка Американскому национальному институту стандартов (ANSI). В 1989-м был опубликован первый стандарт, содержавший ряд усовершенствований, по сравнению с K&R. Во второе издание книги K&R была включена полная спецификация языка, а это означало, что на рабочих столах десятков тысяч программистов появился экземпляр стандарта ANSI [Kernighan 1988]. В 1990 году стандарт ANSI был принят Международной организацией по стандартизации (ИСО) без существенных изменений, но ANSI 89 употребляется чаще (и встречается на футболках).

Прошло десять лет. Язык C стал общепринятым в том смысле, что базовый код практически всех ПК и всех интернет-серверов написан на C; пожалуй, трудно вообразить более «общепринятое» творение человеческого разума.

За это время от C откололся C++ и добился значительного успеха (хотя и не такого значительного, как C). Появление C++ стало лучшим, что когда-либо происходило с C. В то время как другие языки обзаводились дополнительными синтаксическими конструкциями, чтобы следовать в русле объектной ориентированности, и всякими другими фенечками, приходившими на ум их авторам, C строго придерживался стандарта. Те, кто хотел стабильности и переносимости, писали

на C. Те же, кому были нужны все новые и новые возможности, чтобы купаться в них, как в ванне с шампанским, получили в свое распоряжение C++. И все были счастливы.

ISO C99

Спустя десять лет C подвергся существенному пересмотру. Были добавлены средства для численных и научных расчетов, стандартный тип комплексных чисел и некоторые подобия обобщенных функций, адаптирующихся к типу аргументов. Включены некоторые удобные средства C++, включая однострочные комментарии (впервые появившиеся в предшественнике C – языке BCPL) и возможность объявлять переменные в заголовке цикла `for`. Упрощена работа со структурами – благодаря новым правилам объявления и инициализации и некоторым нотационным усовершенствованиям. Признано, что безопасность – не последнее дело и что не все в мире говорят по-английски, и это тоже оказало влияние на модернизацию языка.

Размышляя о том, сколько нового появилось в стандарте C89, и учитывая, что нет в мире компьютера, где бы не работал код на C, трудно представить, что ИСО мог придумать нечто такое, что не подверглось бы ожесточенной критике, – его ругали даже за отказ вносить те или иные изменения. И нельзя не признать, что стандарт оказался противоречивым. Существует два общепринятых способа представить комплексное число (в прямоугольных и в полярных координатах) – так почему ИСО отдал предпочтение только одному? Зачем нужен механизм макросов с переменным числом аргументов, если код прекрасно можно писать и без него? Иными словами, блюстители чистоты идеи обвиняли ИСО в том, что тот уступил давлению со стороны жаждущих новой функциональности. В настоящее время большинство компиляторов поддерживают C99 с некоторыми оговорками, например серьезные трудности вызывает тип `long double`. Однако есть одно заметное исключение из этого широкого консенсуса: корпорация Майкрософт отказывается включать поддержку C99 в свой компилятор Visual Studio C++. В разделе «Компиляция кода на C в Windows» ниже мы увидим некоторые из многочисленных способов откомпилировать код в Windows, так что отказ от Visual Studio – не более чем неудобство, а желание крупного рыночного игрока запретить нам использование стандартов ANSI и ISO только укрепляет дух панк-рока, свойственный C.

C11

Сознавая справедливость обвинений в уступке давлению, ИСО внес серьезные изменения в третью редакцию стандарта. Стало возможно писать обобщенные функции, дальнейшее развитие получили также идеи безопасности и интернационализации.

Стандарт C11 вышел в декабре 2011 года, но разработчики компиляторов на удивление быстро поддержали его. Сейчас большинство основных компиляторов заявляет о почти полном соответствии. Однако стандарт опреде-

ляет поведение не только компилятора, но и стандартной библиотеки, а вот поддержка библиотеки и, в частности, многопоточности и атомарных операций в одних системах реализована, а в других отстает.

Стандарт POSIX

Выше было описано положение дел с самим языком C, однако он всегда развивался вместе с операционной системой Unix, и в книге вы увидите, что эта взаимосвязь существенна для повседневной работы. Если какая-то задача легко решается с помощью командной строки Unix, то, скорее всего, она легко решается и на C; инструментальные средства Unix часто создаются, чтобы упростить кодирование на C.

Unix
C и Unix были разработаны в компании Bell Labs в начале 1970-х годов. На протяжении большей части XX столетия Bell постоянно преследовалась за монополистическую практику, и в одном из соглашений с правительством США компания пообещала не распространять свои коммерческие интересы на разработку программного обеспечения. Таким образом, система Unix была бесплатно отдана исследователям, которые получили право разбирать ее на части и собирать заново. Само слово Unix является торговым наименованием, которое первоначально принадлежало Bell Labs, а затем разошлось, как бейсбольная карточка, среди многочисленных компаний.

Варианты Unix плодились один за другим по мере изучения, переделки и улучшения кода независимыми энтузиастами. Достаточно было одной крохотной несовместимости, чтобы сделать программу или скрипт непереносимыми, поэтому очень скоро была осознана необходимость какой-то стандартизации.

POSIX
Этот стандарт, впервые выпущенный Институтом инженеров электротехники и электроники (IEEE) в 1988 году, заложил общую основу для всех Unix-подобных операционных систем. В нем описывается, как должна работать оболочка, чего ожидать от команд типа `ls` и `grep`, а также ряд библиотек, на которые имеют право рассчитывать программисты, пишущие на C. Например, именно здесь детально описан механизм конвейеров, с помощью которых сцепляются команды в оболочке; это означает, что библиотечная функция `popen` (открыть конвейер) описана в стандарте POSIX, а не ISO C. Стандарт POSIX много раз пересматривался; на момент написания этой книги последней является версия POSIX:2008, и именно ее я имею в виду, говоря, что нечто совместимо с POSIX. В системе, соответствующей стандарту POSIX, должен присутствовать компилятор C, доступный по имени `c99`.
В этой книге стандарт POSIX используется, и я скажу об этом, когда дело до него дойдет. За исключением многочисленного семейства ОС от Майкрософт, практически все прочие операционные системы совместимы с POSIX: Linux, Mac OS X, iOS, webOS, Solaris, BSD, даже в серверных ОС Windows имеется

подсистема POSIX. А о том, как установить подсистему POSIX для систем, выбивающихся из общего ряда, написано в разделе «Компиляция кода на C в Windows» ниже.

Наконец, существуют еще две реализации POSIX, о которых стоит упомянуть ввиду их широкой распространенности и влиятельности.

BSD
После того как компания Bell Labs отдала Unix на растерзание публике, ученые из Калифорнийского университета в Беркли внесли существенные усовершенствования и в конечном итоге полностью переписали код Unix, в результате чего появился дистрибутив Berkeley Software Distribution. Всякий, кто пользуется компьютером производства компании Apple, работает с системой BSD, снабженной привлекательным графическим интерфейсом. В некоторых отношениях BSD выходит за рамки POSIX, и мы увидим функции, которые в стандарт POSIX не входят, но настолько полезны, что обойти их вниманием никак нельзя (и прежде всего такая палочка-выручалочка, как `asprintf`).

GNU
Этот акроним расшифровывается как «GNU is Not Unix», это еще один успешный пример независимой реализации и улучшения окружения Unix. В подавляющем большинстве дистрибутивов Linux используются инструментальные средства GNU. Почти наверняка на вашем POSIX-совместимом компьютере установлен набор компиляторов GNU Compiler Collection (gcc), даже в BSD он есть. Подчеркнем, что gcc – это стандарт де-факто, который в некоторых направлениях расширяет C и POSIX. Всюду, где эти расширения встречаются, я буду упоминать о них явно.

С юридической точки зрения, лицензия BSD чуть более либеральна, чем лицензия GNU. Поскольку некоторые организации глубоко озабочены политическими и деловыми аспектами лицензий, большинство инструментов предлагается в вариантах для GNU и для BSD. Например, как в gcc, так и в BSD имеется проект `clang`, включающий первоклассные компиляторы языка C. Разработчики из обоих лагерей пристально наблюдают друг за другом и перенимают достижения, поэтому можно ожидать, что существующие на сегодня различия в будущем сойдут на нет.

Юридические вопросы

В законодательстве США больше нет системы регистрации прав на интеллектуальную собственность; за немногими исключениями, все, что кем-то написано, автоматически защищено таким правом.
Разумеется, для распространения библиотеки необходимо копирование с одного жесткого диска на другой, и существует целый ряд общеупотребительных механизмов предоставления права копирования произведения, защищенного правом интеллектуальной собственности, с минимумом формальностей.
Лицензия *GNU Public License* разрешает копирование исходного кода и его исполняемого варианта без ограничений. Существует одно важное условие: если вы *распространяете* программу или библиотеку, в которой используется исходный код, защищенный

лицензией GPL, то обязаны распространять и исходный код своей программы. Следует понимать, что если вы используете программу только внутри организации и не распространяете ее, то это условие к вам не относится, и включать в дистрибутив исходный код необязательно. Исполнение программ, распространяемых на условиях GPL, например компиляция своего кода с помощью gcc, не влечет обязательств по распространению своего исходного кода, потому что результат работы программы (например, созданный компилятором исполняемый файл) не считается производным продуктом gcc. [Пример: библиотека GNU Scientific Library.]

Лицензия *Lesser GPL* во многом аналогична GPL, но при этом явно оговаривается, что если вы компонуете свою программу с разделяемой библиотекой, распространяемой на условиях LGPL, то ваш код не считается производным продуктом, и потому распространять исходный код необязательно. Иначе говоря, разрешено распространять без исходного кода программу, скомпонованную с LGPL-библиотекой. [Пример: библиотека GLib.]

Лицензия *BSD* требует сохранения авторских прав и оговорок об ограничении ответственности для распространяемого на условиях этой лицензии исходного кода, но не требует включения исходного кода в дистрибутив. [Пример: библиотека Libxml2, распространяемая по лицензии MIT, аналогичной BSD.]

Примите во внимание обычную оговорку: я не юрист, и это всего лишь краткое изложение довольно длинных юридических документов. Если вы не уверены относительно особенностей своего случая, прочитайте сам документ или обратитесь за консультацией к юристу.

Технические вопросы

Второе издание

Признаться, раньше я был немного циничен и полагал, что второе издание пишут, чтобы подгадить тем, кто собирается продать прочитанный экземпляр первого издания. Но это конкретное второе издание было бы невозможно без выхода первого и не могло бы выйти раньше (да и большинство читателей все равно читает электронные версии).

Самое крупное добавление, по сравнению с первым изданием, – глава о параллельных потоках, иначе говоря, о распараллеливании. Основное внимание в ней уделено библиотеке OpenMP и атомарным переменным и структурам. OpenMP не является частью стандарта C, но входит в экосистему C, и потому ее рассмотрение в этой книге вполне уместно. Атомарные переменные были добавлены в версию стандарта, опубликованную в декабре 2011 года, с этого момента до выхода первого издания даже года не прошло, поэтому в то время ни один компилятор их не поддерживал. Но с тех пор прогресс не стоял на месте, и я смог написать эту главу, опираясь как на изложенную в стандарте теорию, так и на реальный протестированный код (см. главу 12).

Первое издание почтили вниманием некоторые исключительно педантичные читатели. Они заметили все огрехи, которые можно было истолковать как ошибки, – от написанной мной глупости по поводу дефисов в командной строке до неправильно построенных предложений. Ничто в мире не совершенно, но благодаря конструктивным отзывам читателей книга стала гораздо точнее и полезнее.

Перечислю прочие дополнения к первому изданию.

- В приложении А приведен краткий справочник по языку С для читателей, имеющих опыт программирования на других языках. Мне не хотелось включать его в первое издание, потому что учебных пособий по С и так достаточно, но, как выяснилось, с ним книга стала полезнее.
- Откликаясь на часто высказываемое пожелание, я существенно расширил материал по работе с отладчиком (см. раздел «Работа с отладчиком»).
- В первом издании был раздел о том, как писать функции с переменным числом аргументов, так чтобы оба вызова `sum(1, 2.2)` и `sum(1, 2.2, 3, 8, 16)` были правильны. Но что, если требуется передать несколько списков, например вычислить скалярное произведение двух векторов произвольной длины: `dot((2, 4), (-1, 1))` и `dot((2, 4, 8, 16), (-1, 1, -1, 1))` (см. раздел «Несколько списков»)?
- Я переписал главу 11 о расширении объектов путем добавления новых функций. Основное дополнение – реализация виртуальных таблиц.
- Я немного дополнил материал о препроцессоре, уделив внимание трудному вопросу о тестовых макросах, включая и мимолетное упоминание ключевого слова `_Static_assert`.
- Я остался верен данному самому себе обещанию не включать в эту книгу справочный материал по регулярным выражениям (потому что в сети и в других книгах в этом нет недостатка). Но все же добавил демонстрационный пример в разделе «Разбор регулярных выражений», посвященном использованию определенных в POSIX функций для работы с регулярными выражениями, которые, по сравнению с другими языками, являются довольно низкоуровневыми.
- Обсуждение работы со строками в первом издании во многом опиралось на функцию `asprintf`, которая похожа на `sprintf`, но автоматически выделяет необходимое количество памяти перед записью в нее строки. Существует версия этой функции в дистрибутиве GNU, но многие читатели не могут ей воспользоваться из-за условий лицензии, поэтому в примере 9.3 в этом издании я показал, как написать такую функцию, пользуясь только стандартными конструкциями С.
- Один из серьезных вопросов, обсуждаемых в главе 7, – тот факт, что детальное управление числовыми типами может приводить к проблемам, поэтому в первом издании я не упомянул о десятках новых числовых типов, появившихся в стандарте С99, например: `int_least32_t`, `uint_fast64_t` и т. д. (С99 §7.18; С11 §7.20). Несколько читателей настойчиво просили меня отметить хотя бы наиболее полезные типы, например `intptr_t` и `intmax_t`, что я и делаю в подходящем месте.

Графические выделения

В книге применяются следующие графические выделения:

Курсив
Новые термины, URL-адреса, адреса электронной почты, имена и пути к файлам. Многие термины определены в глоссарии в конце книги.

`Моноширинный`
Листинги программ, а также элементы кода в основном тексте: имена переменных и функций, базы данных, типы данных, переменные окружения, предложения и ключевые слова языка.

`Моноширинный курсив`
Текст, вместо которого следует подставить значения, заданные пользователем или определяемые контекстом.


Так обозначается совет, рекомендация или замечание общего характера.


Так обозначаются упражнения, призванные помочь в освоении.


Так обозначается предупреждение или предостережение.

О примерах кода

Эта книга призвана помогать вам в работе. Поэтому вы можете использовать приведенный в ней код в собственных программах и в документации. Спрашивать у нас разрешение необязательно, если только вы не собираетесь воспроизводить значительную часть кода. Например, никто не возбраняет включить в свою программу несколько фрагментов кода из книги. Однако для продажи или распространения примеров из книг издательства O'Reilly на компакт-диске разрешение требуется. Цитировать книгу и примеры в ответах на вопросы можно без ограничений. Но для включения значительных объемов кода в документацию по собственному продукту нужно получить разрешение.

Примеры кода можно скачать с сайта **https://github.com/b-k/21st-Century-Examples**.

Мы высоко ценим, хотя и не требуем, ссылки на наши издания. В ссылке обычно указываются название книги, имя автора, издательство и ISBN, например: «21st Century C by Ben Klemens (O'Reilly). Copyright 2013 Ben Klemens, 978-1-449-32714-9».

Если вы полагаете, что планируемое использование кода выходит за рамки изложенной выше лицензии, пожалуйста, обратитесь к нам по адресу **permissions@oreilly.com**.

Как с нами связаться

Вопросы и замечания по поводу этой книги отправляйте в издательство:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (в США и Канаде)
707-829-0515 (международный или местный)
707-829-0104 (факс)

Для этой книги имеется веб-страница, на которой публикуются списки замеченных ошибок, примеры и прочая дополнительная информация. Адрес страницы: **http://oreil.ly/21st_century_c**.

Замечания и вопросы технического характера следует отправлять по адресу **bookquestions@oreilly.com**.

Дополнительную информацию о наших книгах, конференциях и новостях вы можете найти на нашем сайте по адресу **http://www.oreilly.com**.

Читайте нас на Facebook: **http://facebook.com/oreilly**.

Следите за нашей лентой в Twitter: **http://twitter.com/oreillymedia**.

Смотрите нас на YouTube: **http://www.youtube.com/oreillymedia**.

Благодарности

- **Нора Альберт**: общая поддержка, подопытный кролик.
- **Брюс Филдс**, **Дэйв Китабян**, **Сара Вейссман**: скрупулезное рецензирование.
- **Патрик Холл**: знание Unicode.
- **Натан Джепсон**, **Эллисон Макдональд**, **Рейчел Румелиотис**, **Шон Уоллес**: редактирование.
- **Андреас Клейн**: указание на ценность типа `intptr_t`.
- **Роландо Родригес**: тестирование, любознательное использование, вдумчивое исследование.
- **Рейчел Стили**: производство.
- **Ульрик Свердруп**: указание на то, как использовать позиционные инициализаторы для задания значений по умолчанию.