Содержание

Эта книга посвящается
Жасмин Бланшетт (Jasmin Blanchette) и Трентону Шульцу (Trenton Schulz)

Введение

Цель этой книги – научить специфике программирования на языке Go с использованием всех его характерных особенностей, а также рассказать о наиболее часто применяемых пакетах, входящих в состав стандартной библиотеки Go. Книга также задумывалась как справочник для тех, кто уже знаком с языком. Чтобы соответствовать этим двум целям, была сделана попытка охватить сразу все темы в одной книге, и в текст были добавлены перекрестные ссылки.

По духу Go подобен языку C – это компактный и эффективный язык программирования с низкоуровневыми возможностями, такими как указатели. Однако Go обладает множеством особенностей, характерных для высоко- и очень высокоуровневых языков, таких как поддержка строк Юникода, высокоуровневые структуры данных, динамическая типизация, автоматическая сборка мусора и высокоуровневая поддержка взаимодействий, основанных на обмене сообщениями, а не на блокировках и разделяемых данных. Кроме того, язык Go имеет обширную и всестороннюю стандартную библиотеку.

Предполагается, что читатель уже имеет опыт программирования на распространенных языках, таких как C, C++, Java, Python и им подобных, тем не менее все уникальные особенности и идиомы языка Go демонстрируются на законченных, работающих примерах, подробно описываемых в тексте.

Для успешного освоения любого языка программирования совершенно необходимо писать программы на этом языке. С этой точки зрения в книге предпринят абсолютно практический подход: читателям предлагается не бояться экспериментировать с примерами, выполнять предлагаемые упражнения и писать собственные программы, чтобы обрести практический опыт. Как и во всех моих предыдущих книгах, все фрагменты программного кода являются «живым кодом», то есть этот код автоматически извлекался из исходных файлов `.go` и вставлялся в документ PDF перед передачей издателю, поэтому все примеры гарантированно работоспособны и

в них исключены ошибки, возможные при копировании вручную. Везде, где только возможно, в качестве примеров демонстрируются небольшие, но законченные программы и пакеты. Все примеры, упражнения и решения доступны на сайте www.qtrac.eu/gobook.html.

Основной целью книги является обучение *языку* Go, здесь рассказывается о многих пакетах из стандартной библиотеки Go, но далеко не обо всех. Однако в этом нет никакой проблемы, поскольку книга дает достаточный объем информации о Go, чтобы читатель смог самостоятельно использовать любые стандартные или сторонние пакеты и конечно же создавать собственные.

Зачем изучать язык Go?

Разработка языка Go началась в 2007 году как внутренний проект компании Google. Оригинальная архитектура языка была разработана Робертом Гризмером (Robert Griesemer) и корифеями ОС Unix – Робом Пайком (Rob Pike) и Кеном Томпсоном (Ken Thompson). 10 ноября 2009 года были опубликованы исходные тексты реализации языка Go под либеральной открытой лицензией. Развитие языка Go продолжается группой разработчиков из компании Google, в состав которой входят основатели языка, а также Расс Кокс (Russ Cox), Эндрю Джерранд (Andrew Gerrand), Ян Ланс Тейлор (Ian Lance Taylor) и многие другие. Разработка ведется с использованием открытой модели, благодаря чему в процессе участвуют многие разработчики со всего мира, порой настолько известные и уважаемые, что им предоставлены те же привилегии доступа к репозиторию с исходными текстами, что и специалистам из компании Google. Кроме того, в общественном доступе имеется множество сторонних пакетов для языка Go, доступных на сайте Go Dashboard (godashboard.appspot.com/project).

Go – один из самых удивительных языков, появившихся в последние 15 лет, и первый, нацеленный на программистов и компьютеры XXI века.

Go проектировался с прицелом на эффективное масштабирование, благодаря чему его можно использовать для создания очень больших приложений и компиляции даже очень больших программ за секунды на единственном компьютере. Молниеносная скорость компиляции обеспечивается отчасти простотой синтаксического анализа программ на этом языке, но главным образом благодаря особенностям управления зависимостями. Например, если файл `app.`

go зависит от файла pkg1.go, который, в свою очередь, зависит от файла pkg2.go, в обычных компилирующих языках для компиляции файла app.go необходимо иметь объектные модули, полученные в результате компиляции обоих файлов, pkg1.go и pkg2.go. Но в Go все, что экспортирует pkg2.go, включено в объектный модуль для файла pkg1.go, поэтому для компиляции app.go достаточно иметь только объектный модуль для файла pkg1.go. Для случая с тремя файлами это едва ли имеет большое значение, но в огромных приложениях с большим количеством зависимостей эта особенность дает весьма значительный прирост скорости компиляции.

Благодаря высокой скорости компиляции программ на языке Go появляется возможность использовать этот язык в областях, где обычно применяются языки сценариев (см. врезку «Сценарии на языке Go» ниже). Кроме того, язык Go можно использовать для создания веб-приложений с применением Google App Engine.

Язык Go имеет очень простой и понятный синтаксис, в котором отсутствуют сложные и замысловатые конструкции, характерные для более старых языков, таких как C++ (появившегося в 1983 году) или Java (появившегося в 1995 году). И относится к категории языков со строгой статической типизацией, что многими программистами считается важным условием для разработки крупных программ. Однако система типов данных в языке Go не слишком обременительна благодаря поддержке синтаксиса объявления переменных одновременно с их инициализацией (когда компилятор определяет тип автоматически, избавляя от необходимости явно указывать его) и наличию мощного и удобного механизма динамической типизации.

Языки программирования, такие как C и C++, требуют от программистов выполнения массы работы, когда дело доходит до управления памятью, которую можно переложить на плечи компьютера, особенно в многопоточных приложениях, где учет использования динамической памяти может оказаться невероятно сложной задачей. В последние годы ситуация в этой области в языке C++ намного улучшилась благодаря появлению «интеллектуальных» указателей, но он пока не способен догнать язык Java с его библиотекой поддержки многопоточной модели выполнения. Язык Java освобождает программиста от бремени управления памятью с помощью механизма сборки мусора. Поддержка многопоточной модели выполнения в языке C++ в настоящее время включена в состав стандартной библиотеки, однако в языке C она реализована только в виде сторонних библиотек. Но, несмотря на все это, создание многопоточных

программ на языке C, C++ или Java требует от программиста немалых усилий, чтобы обеспечить своевременное приобретение и освобождение ресурсов.

Все сложности, связанные с учетом ресурсов в языке Go, берут на себя компилятор и среда выполнения. Для управления памятью в Go имеется механизм сборки мусора, что избавляет от необходимости использовать «интеллектуальные» указатели или освобождать память вручную. А поддержка параллелизма в языке Go реализована в форме механизма взаимодействующих последовательных процессов (Communicating Sequential Processes, CSP), основанного на идеях специалиста в области теории вычислительных машин и систем Чарльза Энтони Ричарда Хоара (C. A. R. Hoare), благодаря которому во многих многопоточных программах на языке Go вообще отпадает необходимость блокировать доступ к ресурсам. Кроме того, в языке Go имеются так называемые *go-подпрограммы* (*goroutines*) – очень легковесные процессы, которых можно создать великое множество. Выполнение этих процессов автоматически будет распределяться по доступным процессорам и ядрам, что обеспечивает возможность более тонкого деления программ на параллельно выполняющиеся задачи, чем это позволяют другие языки программирования, основанные на потоках выполнения. Фактически поддержка параллелизма в языке Go реализована настолько просто и естественно, что при переносе однопоточных программ на язык Go часто обнаруживается возможность параллельного выполнения нескольких задач, ведущая к увеличению скорости выполнения и более оптимальному использованию машинных ресурсов.

Go – практичный язык, где во главу угла поставлены эффективность программ и удобство программиста. Например, встроенные и определяемые пользователем типы данных в языке Go существенно отличаются – операции с первыми из них могут быть значительно оптимизированы, что невозможно для последних. В Go имеются также два встроенных фундаментальных типа коллекций: *срезы* (*slices*) (фактически ссылки на массивы переменной длины) и *отображения* (*maps*) (словари, или хеши пар ключ/значение). Коллекции этих типов высокооптимизированы и с успехом могут использоваться для решения самых разных задач. В языке Go также поддерживаются указатели (это действительно компилирующий язык программирования – в нем отсутствует какая-либо виртуальная машина, снижающая производительность), что позволяет с непринужденностью

создавать собственные, весьма сложные типы данных, такие как сбалансированные двоичные деревья.

В то время как C поддерживает только процедурное программирование, а Java вынуждает программистов писать все программы в объектно-ориентированном стиле, Go позволяет использовать парадигму, наиболее подходящую для конкретной задачи. Go можно использовать как исключительно процедурный язык программирования, но он также обладает превосходной поддержкой объектно-ориентированного стиля программирования. Однако, как будет показано далее в книге, реализация объектно-ориентированной парадигмы в Go радикально отличается от реализации этой же парадигмы в таких языках, как C++, Java или Python. Она намного проще в использовании и значительно гибче.

Как и в языке C, в Go отсутствуют генерики (generics) (шаблоны, в терминологии C++), однако в Go имеется масса других возможностей, которые во многих случаях устраняют потребность в генериках. В языке Go отсутствует препроцессор и не используются подключаемые заголовочные файлы (что является еще одной причиной, объясняющей высокую скорость компиляции), поэтому в нем нет необходимости дублировать сигнатуры функций, как в C и C++. А благодаря отсутствию препроцессора семантика программы не может измениться незаметно для программиста, как это может произойти при небрежном обращении с директивами `#define` в языках C и C++.

Возможно, языки C++, Objective-C и Java создавались как улучшенные версии языка C (а последний – как улучшенная версия языка C++). В языке Go тоже можно усмотреть попытку создать улучшенную версию языка C, даже при том, что простой и ясный синтаксис Go больше напоминает язык Python – срезы и отображения в Go сильно напоминают списки и словари в Python. Однако по духу Go все-таки ближе к языку C, чем к любым другим языкам, и его можно считать попыткой устранить недостатки языка C, взять из него все самое лучшее и добавить множество новых возможностей, уникальных для Go.

Первоначально Go задумывался как язык системного программирования с высокой скоростью компиляции для разработки высокомасштабируемых программ, которые могли бы использовать преимущества распределенных систем и многоядерных компьютеров. В настоящее время область применения языка Go стала значительно шире первоначальной концепции, и сейчас он используется как высокопроизводительный язык программирования общего назначения, использовать который – одно удовольствие.

Структура книги

Глава 1 начинается с описания, как компилировать и запускать программы на языке Go. Затем в этой главе дается краткий обзор синтаксиса и возможностей языка Go, а также вводятся некоторые пакеты из стандартной библиотеки. В ней будут представлены и описаны пять коротких примеров, иллюстрирующих различные возможности языка Go. Эта глава написана так, чтобы сформировать представление о языке и вселить в читателя уверенность в необходимости освоения языка Go. (В этой главе также описывается, как получить и установить Go.)

Главы со 2 по 7 детально рассматривают язык Go. Три главы посвящены встроенным типам данных: в главе 2 рассказывается об идентификаторах, логических и числовых типах. В главе 3 рассматриваются строки. И в главе 4 – коллекции.

Глава 5 описывает и демонстрирует инструкции и управляющие конструкции языка Go. Она также рассказывает, как создавать и использовать собственные функции, и заканчивает главы, демонстрирующие создание на языке Go однопоточных программ в процедурном стиле.

Глава 6 показывает особенности объектно-ориентированного программирования на языке Go. Данная глава включает описание структур языка Go, используемых для объединения и встраивания (делегирования) значений, и интерфейсов, применяемых для определения абстрактных типов, а также демонстрирует, как в некоторых ситуациях добиться эффекта наследования. В главе будут представлены несколько законченных примеров с подробным описанием, чтобы помочь читателю разобраться в объектно-ориентированном стиле программирования на языке Go, который может существенно отличаться от привычного стиля.

Глава 7 охватывает механизмы параллельного выполнения задач в языке Go и приводит еще больше примеров, чем глава об объектно-ориентированном программировании, опять же чтобы помочь читателю лучше понять эти новые аспекты.

Глава 8 демонстрирует, как читать из файлов и записывать в них собственные и стандартные двоичные данные, текст, а также данные в формате JSON и XML. (Работа с текстовыми файлами коротко рассматривается в главе 1 и в нескольких последующих главах, потому что это позволяет приводить более практичные примеры и упражнения.)

Глава 9 завершает книгу. Она начинается с демонстрации импортирования и использования пакетов из стандартной библиотеки, а затем собственных и сторонних пакетов. В ней также рассказывается,

как документировать и тестировать собственные пакеты, измерять их производительность. Последний раздел главы является кратким обзором инструментов, предоставляемых компилятором `gc` и стандартной библиотекой Go.

Go – небольшой, но очень богатый и выразительный язык программирования (если измерять количеством синтаксических конструкций, концепций и идиом), поэтому книга получилась такой удивительно объемной. В ней с самого начала демонстрируются примеры, написанные в хорошем стиле, характерном для языка Go[1]. Разумеется, что при таком подходе некоторые приемы сначала демонстрируются и лишь потом разъясняются подробно. Читатель может быть уверен, что все необходимое обязательно будет разъясняться в книге (и, разумеется, будут даваться ссылки на пояснения, находящиеся в другом месте).

Go – очаровательный язык, и пользоваться им доставляет одно удовольствие. Синтаксис и идиомы языка просты в изучении, но в нем имеются несколько совершенно новых концепций, которые могут оказаться незнакомыми многим читателям. Данная книга пытается помочь читателю совершить концептуальный прорыв, особенно в области объектно-ориентированного и параллельного программирования на языке Go, что может занять недели, а то и месяцы у тех, в чьем распоряжении имеется только сухая документация.

Благодарности

Ни одна техническая книга, написанная мной, не обошлась без помощи других людей, и эта книга также не является исключением.

Я хотел бы выразить особую благодарность двум моим друзьям-программистам, не имевшим прежде опыта работы с языком Go: Жасмин Бланшетт (Jasmin Blanchette) и Трентону Шульцу (Trenton Schulz). Оба они на протяжении многих лет помогали мне в создании книг, и их отзывы помогли мне написать книгу, которая отвечала бы потребностям других программистов, начинающих изучение языка Go.

Также книга существенно выиграла благодаря отзывам одного из основных разработчиков Go – Найджела Тао (Nigel Tao). Я не всегда следовал его советам, но его мнение всегда имело большое значение и

[1] Единственное исключение составляют примеры использования каналов, которые в первых главах всегда объявляются как двунаправленные, хотя используются для передачи данных только в одном направлении. Корректное объявление направления обмена данными в каналах производится с главы 7.

помогло значительно улучшить не только примеры программного кода, но и текст.

Кроме того, я получал помощь и от других, включая Дэвида Бодди (David Boddie), начинающего программиста на языке Go, давшего ряд ценных советов. А также от разработчиков Go: Яна Ланса Тейлора (Ian Lance Taylor) и особенно от Расса Кокса (Russ Cox), решившего немало проблем с программным кодом и пониманием концепций, и давшего простые и ясные пояснения, способствовавшие повышению точности изложения технических деталей.

В процессе работы над книгой я неоднократно задавал вопросы в списке рассылки `golang-nuts` и всегда получал вдумчивые и полезные ответы от многих собеседников. Я также получал отзывы от читателей предварительного, «чернового» издания, опубликованного на сайте Safari Book Online, что повлекло добавление важных пояснений.

Итальянская компания (www.develer.com), занимающаяся разработкой программного обеспечения, в лице Джованни Баджо (Giovanni Bajo) любезно предоставила бесплатный хостинг для хранения репозитория Mercurial, обеспечив мне душевное спокойствие в течение длинного периода работы над этой книгой. Спасибо Лоренцо Манцини (Lorenzo Mancini), настроившему и сопровождавшему этот репозиторий для меня. Я также очень благодарен Антону Бауэрсу (Anton Bowers) и Бену Томпсону (Ben Thompson), предоставившим мне место для моего веб-сайта www.qtrac.eu на их веб-сервере в начале 2011 года.

Спасибо Расселу Уиндеру (Russel Winder) за статью о лицензиях на программное обеспечение, размещенную в его блоге www.russel.org.uk. Многие его идеи были заимствованы в приложении B.

И, как обычно, спасибо Джеффу Кингстону (Jeff Kingston), создателю неуклюжей типографской системы, которую я на протяжении многих лет использовал для набора всех своих книг и многих других трудов.

Особое спасибо моему выпускающему редактору Дебре Уильямс Коли (Debra Williams Cauley), успешно подготовившей книгу совместно с издателем и обеспечившей техническую поддержку и практическую помощь.

Спасибо также заведующему производственным отделом Анне Попик (Anna Popick), которая снова с успехом обеспечила руководство технологическим процессом, а также корректору Одри Дойл (Audrey Doyle), великолепно справившейся со своей работой.

Как всегда, я хочу поблагодарить мою супругу Андреа (Andrea) за ее любовь и поддержку.

1. Обзор в пяти примерах

В этой главе приводится серия из пяти примеров с подробными их описаниями. Несмотря на небольшой размер примеров, каждый из них (кроме «Hello Who?») имеет некоторую практическую ценность, а все вместе они представляют краткий обзор ключевых возможностей языка Go и некоторых его основных пакетов. (То, что в других языках называется модулями или библиотеками, в языке Go называется *пакетами*, а все пакеты, распространяемые вместе с Go, образуют *стандартную библиотеку* языка Go.) Цель этой главы – сформировать представление о языке и вселить в читателя уверенность в необходимости освоения языка Go. Не стоит волноваться, если какие-то синтаксические конструкции или идиомы останутся за рамками понимания, – все, что будет показано в этой главе, обязательно будет рассматриваться в последующих главах.

Обучение программированию на языке Go с применением специфических приемов требует определенного времени и усилий. Желающие заняться переносом программ с языков C, C++, Java, Python и др. на язык Go должны найти время на изучение Go, особенно его объектно-ориентированных возможностей и средств организации параллельных вычислений, в долгосрочной перспективе это позволит сэкономить время и силы. А желающие писать на языке Go собственные приложения добьются большего успеха, если максимально будут использовать все его возможности, поэтому они также должны потратить силы и время на изучение – позднее все затраты окупятся с лихвой.

1.1. Начало

Go – это компилирующий язык, а не интерпретирующий, поэтому программы, написанные на этом языке, имеют максимальную производительность. Компиляция выполняется очень быстро – намного быстрее, чем в некоторых других языках, особенно в сравнении с языками C и C++.

Документация по языку Go

По адресу golang.org находится официальный веб-сайт языка Go, где можно найти массу свежей документации по языку Go. По ссылке **Packages** (Пакеты) можно перейти к разделу с документацией ко всем пакетам из стандартной библиотеки Go и их исходными текстами, которые могут очень пригодиться, когда в документации обнаруживаются пробелы. Ссылка **References** => **Command Documentation** (Справочники => Документация к командам) ведет в раздел с документацией к программам, распространяемым вместе с Go (компиляторам, инструментам сборки и др.). По ссылке **References** => **Language Specification** (Справочники => Спецификация языка) можно перейти к разделу с неофициальной и весьма полной спецификацией языка Go. А по ссылке **Documents => Effective Go** (Документы => Эффективный Go) находится документ, описывающий многие приемы программирования на Go.

На веб-сайте также имеется «песочница» (с ограниченным набором возможностей), где можно вводить, компилировать и опробовать небольшие программы на языке Go. Данная возможность будет полезна начинающим для проверки непонятных синтаксических конструкций и изучения сложного пакета `fmt`, содержащего инструменты для работы с форматированным текстом, или пакета `regexp` – механизма регулярных выражений. Строка поиска на веб-сайте языка Go может использоваться только для поиска документации – если потребуется отыскать другие ресурсы, посвященные языку Go, посетите страницу go-lang.cat-v.org/go-search.

Документацию по языку Go можно также просматривать локально, например в веб-браузере. Для этого выполните команду `godoc`, передав ей аргумент, сообщающий, что она должна действовать как веб-сервер. Ниже показано, как выполнить эту команду в консоли Unix (`xterm`, `gnome-terminal`, `konsole`, `Terminal.app` и др.):

```
$ godoc -http=:8000
```

Или в консоли Windows (например, в окне **Command Prompt** (Командная строка) или **MS-DOS Prompt** (Сеанс MS-DOS)):

```
C:\>godoc -http=:8000
```

Номер порта здесь выбран произвольно. Если он уже занят – просто выберите другой. Здесь предполагается, что выполняемый файл `godoc` находится в каталоге, указанном в переменной `PATH`.

Для просмотра локальной документации откройте веб-браузер и введите адрес http://localhost:8000. На экране появится страница, внешним видом напоминающая главную страницу веб-сайта golang.org. Ссылка **Packages** (Пакеты) ведет в раздел документации к стандартной библиотеке Go, где также имеется документация к сторонним

пакетам, установленным в каталог GOROOT. Если в системе определена переменная окружения GOPATH (например, для локальных программ и пакетов), рядом со ссылкой **Packages** (Пакеты) появится ссылка к разделу с соответствующей документацией. (Переменные GOROOT и GOPATH обсуждаются ниже в этой главе, а также в главе 9.) С помощью команды godoc можно еще просматривать документацию для всего пакета в целом или для отдельного его элемента непосредственно в консоли. Например, команда godoc image NewRGBA выведет описание функции image.NewRGBA(), а команда godoc image/png – описание пакета image/png в целом.

Стандартный компилятор языка Go называется gc, а в состав его инструментов входят программы: 5g, 6g и 8g – для компиляции, 5l, 6l и 8l – для компоновки и godoc – для просмотра документации. (В Windows эти программы называются 5g.exe, 6l.exe и т. д.) Такие странные имена были даны в соответствии с соглашениями об именовании компиляторов, принятыми в операционной системе Plan 9, где цифра определяет аппаратную архитектуру (например, «5» – ARM, «6» – AMD-64, включая 64-битные процессоры Intel, и «8» – Intel 386.) К счастью, нет необходимости напрямую использовать эти инструменты благодаря наличию высокоуровневого инструмента сборки программ на языке Go – go, который автоматически выбирает нужный компилятор и компоновщик.

Все примеры из этой книги, доступные для загрузки на странице www.qtrac.eu/gobook.html, были проверены в Linux и Mac OS X с помощью gc, и в Windows с помощью компилятора версии Go 1. Разработчики Go предполагают обеспечить обратную совместимость с версией Go 1 во всех последующих версиях Go 1.x, поэтому описание в книге и примеры должны быть верными для всей серии 1.x. (Со временем, при обнаружении каких-либо несовместимостей, загружаемые примеры для книги будут обновляться в соответствии с последней версией Go, поэтому они могут отличаться от программного кода в книге.)

Чтобы загрузить и установить Go, откройте страницу golang.org/doc/install.html, где приводятся ссылки для загрузки и инструкции по установке. На момент написания этих строк версия Go 1 была доступна в виде двоичных и исходных файлов для FreeBSD 7+, Linux 2.6+, Mac OS X (Snow Leopard и Lion) и Windows 2000+ и во всех случаях для аппаратных архитектур Intel 386 and AMD-64. Для Linux имеется также поддержка архитектуры ARM. Предварительно собранные пакеты Go имеются для дистрибутива Ubuntu Linux, и

к моменту, когда вы будете читать эти строки, они могут появиться для других дистрибутивов Linux. Для начинающих изучать программирование на языке Go проще установить двоичную версию, чем собирать инструменты Go из исходных текстов.

Для программ, собираемых компилятором gc, действуют определенные соглашения об именовании. То есть программы, скомпилированные с помощью gc, могут быть скомпонованы только с внешними библиотеками, следующими тем же соглашениям, в противном случае необходимо использовать подходящий инструмент, устраняющий разногласия. В комплект Go входит инструмент cgo (golang.org/cmd/cgo), обеспечивающий возможность использования внешнего программного кода на языке C в программах на языке Go, кроме того, в Linux и BSD-системах имеется возможность использовать код на C и C++ с помощью инструмента SWIG (www.swig.org).

Помимо gc, имеется также компилятор gccgo. Это интерфейс к компилятору gcc (GNU Compiler Collection) для языка Go, который может быть задействован с компиляторами gcc, начиная с версии 4.6. Подобно gc, компилятор gccgo может быть доступен в некоторых дистрибутивах Linux в виде готовых пакетов. Инструкции по сборке и установке компилятора gccgo можно найти на странице golang.org/doc/gccgo_install.html.

1.2. Правка, компиляция и запуск

Программы на языке Go записываются в виде простого текста Юникода с использованием кодировки UTF-8[1]. Большинство современных текстовых редакторов обеспечивают эту поддержку автоматически, а некоторые наиболее популярные из них поддерживают даже подсветку синтаксиса для языка Go и автоматическое оформление отступов. Если ваш текстовый редактор не поддерживает Go, попробуйте ввести имя редактора в строке поиска на сайте Go, чтобы узнать, имеются ли для него расширения, обеспечивающие требуемую поддержку. Для удобства правки все ключевые слова и операторы языка Go записываются символами ASCII, однако идентификаторы в языке Go могут начинаться с любых алфавитных символов

[1] Некоторые текстовые редакторы для Windows (такие как Notepad (Блокнот)) не следуют рекомендациям стандарта Юникода и вставляют байты 0xEF, 0xBB, 0xBF в начало файлов с текстом в кодировке UTF-8. В этой книге предполагается, что файлы в кодировке UTF-8 не содержат этих байтов.

Юникода и содержать любые алфавитные символы Юникода или цифры. Благодаря этому программисты на Go свободно могут определять идентификаторы на своем родном языке.

Сценарии на языке Go

Одним из побочных эффектов высокой скорости компиляции программ на языке Go является возможность создания сценариев в Unix-подобных системах, начинающихся со строки `#!`. Для этого достаточно лишь установить подходящий инструмент, выполняющий компиляцию и запуск программы. На момент написания этих строк имелись два таких инструмента: `gonow` (github.com/kless/gonow) и `gorun` (wiki.ubuntu.com/gorun).

После установки `gonow` или `gorun` любую программу на языке Go можно оформить в виде сценария. Достигается это выполнением двух простых действий. Первое – добавить строку `#!/usr/bin/env gonow` или `#!/usr/bin/env gorun` в самое начало файла с расширением `.go`, содержащим функцию `main()` (в пакете `main`). Второе – дать файлу права на выполнение (например, командой `chmod +x`). Такие файлы могут компилироваться только инструментами `gonow` и `gorun`, потому что строка `#!` не является синтаксически допустимой строкой на языке Go.

При первом запуске команда `gonow` или `gorun` скомпилирует файл с расширением `.go` (очень быстро, разумеется) и запустит его. При последующих попытках перекомпиляция будет выполняться, только если исходный файл `.go` изменился с момента предыдущей компиляции. Это делает возможным написание на языке Go различных небольших вспомогательных программ, например для решения задач системного администрирования.

Чтобы получить представление, как писать, компилировать и выполнять программы на языке Go, начнем с классического примера «Hello World». Несмотря на небольшой размер, программа будет выглядеть чуть сложнее, чем обычно. Но для начала обсудим компиляцию и запуск программы, а затем, в следующем разделе, перейдем к детальному изучению исходного программного кода в файле `hello/hello.go`, использующего некоторые базовые идеи и особенности языка Go.

Все примеры для этой книги доступны на странице www.qtrac.eu/gobook.html в виде архива каталога `goeg`. Поэтому полный путь к файлу `hello.go` (предполагается, что архив с примерами распакован непосредственно в домашний каталог, хотя его можно распаковать в любой другой каталог) будет иметь вид `$HOME/goeg/src/hello/hello.go`. При ссылке на имена файлов в этой книге всегда