

der einfache Wechsel in das Go-Quellverzeichnis und der Start des Übersetzungsskripts.

```
$ cd $HOME/go/src
$ ./all.bash
```

Neben der Übersetzung der Programme werden auch die mitgelieferten Tests ausgeführt. Das Ende der Ausgabe hat typischerweise diese Form:

```
N known bugs; 0 unexpected bugs
```

Die durch N repräsentierte Anzahl an Fehlern kann dabei von Release zu Release schwanken und ist kein Anlass zur Sorge. Sie sind dem Entwicklerteam bekannt und werden nach und nach behoben.

Aktuell erscheinen etwa alle zwei Wochen neue Releases, die Entwicklung ist noch stürmisch. Sie werden auf der Website, via Mailingliste und Twitter angekündigt. Hierzu in Kapitel 7 mehr. Die Aktualisierung auf den neusten Stand ist dabei unkompliziert. Es genügt die Befehlsfolge

```
$ cd $HOME/go/src
$ hg pull
$ hg update release
$ ./all.bash
```

1.4 Das erste Programm

Der erste Berührungspunkt mit einer Vielzahl von Programmiersprachen ist das *Hello-world-Programm*, das in kurzer und knapper Form die Struktur eines Programms demonstriert². So wollen wir es nun auch mit Go halten.

```
// Hello, world.
package main

import "fmt"

func main() {
    var world string = "world"

    fmt.Printf("Hello, %v!\n", world)
}
```

Quelltext 1.2

»Hello, world« in Go

²Bekannt geworden ist es mit dem Buch *The C Programming Language* von Brian Kernighan und Dennis Ritchie. Heute liegt es in nahezu allen Programmiersprachen vor.

Nach einem dokumentierenden Zeilenkommentar am Anfang – er wird durch zwei Slashes (`//`) eingeleitet – zeigt das Beispiel, wie Programme in Go mit dem Schlüsselwort **package** in *Packages* organisiert sind. Diese sind in der Regel Bibliotheken, also Mengen von Funktionen, Typen, Konstanten und Variablen, die anderen Programmen zur Nutzung zur Verfügung gestellt werden. Unser Beispielprogramm importiert das Package `fmt` für die formatierte Ein- und Ausgabe von Texten.

Der Packagename für das ausführbare Hauptprogramm selbst ist fest vorgegeben und lautet immer `main`, ebenso wie die Hauptfunktion den Namen `main()` trägt. Ihre Deklaration wird durch das Schlüsselwort **func** eingeleitet. Sie darf nur einmal enthalten sein, hat keine Parameter und auch keinen Rückgabewert. Mit dem Ende der Funktion wird auch das Programm beendet.

Wird ein Zugriff auf Aufrufparameter benötigt, so stellt Go hierfür das Package `flag` zur Verfügung. Es bietet Funktionen für die automatische Analyse der Parameter sowie für die Definition von Hilfstexten für diese. Gleichzeitig kann ein Programm an beliebiger Stelle beendet werden. Hierfür bietet Go im Package `os` die Funktion `Exit(code int)`. Sie erlaubt zudem die Angabe eines Rückgabewertes für das Programm.

Innerhalb der Hauptfunktion wird nun über das Schlüsselwort **var** zuerst die Variable `world` vom Typ **string** deklariert. Dieser Typ dient der Aufnahme von Strings. Im Beispiel wird der Variablen so noch während der Deklaration der String `"world"` zugewiesen. Für diese gleichzeitige Deklaration mit Initialisierung bietet Go eine praktische Abkürzung, hierzu später mehr. Für die interne Kodierung von Bezeichnern und Strings nutzt Go *UTF-8*. Somit sind internationale Anwendungen jenseits der englischen Sprache problemlos zu realisieren.

Die Ausgabe eines Strings auf der Standardausgabe erfolgt über die Funktion `Printf()` aus dem bereits oben erwähnten Package `fmt`. Der erste Parameter ist eine Formatvorlage, die Platzhalter für einzufügende Elemente enthalten kann. Diese werden durch die weiteren optionalen Parameter des Funktionsaufrufs bestimmt. Hier im Beispiel ist es die zuvor deklarierte und initialisierte Variable `world`. Beim Aufruf der Funktion `Printf()` wird ihrem Namen der Name des Packages vorangestellt, das sie implementiert. Dies dient der Abgrenzung von gleichnamigen Bezeichnern aus dem eigenen oder anderen importierten Packages.

Das Ergebnis unserer Bemühungen ist nun ein kurzes Programm, das aus einem Format-String und der Variablen den endgültigen String erstellt und diesen auf der Standardausgabe druckt: *Hello, world!*