Go Programlama

Murat Özalp

İçindekiler

1

Go Nasıl Bir Programlama Dilidir?

Bu Bölümde

Bu bölümde, Google'ın neden bu dili desteklediği, Go'nun hedef kitlesinin hangi tür ihtiyaçları olduğu, hangi programcılara hitap ettiği gibi soruların cevabını vermeye çalışacağız.

GIRIŞ

Piyasada bu kadar fazla programlama dili varken, Go'ya ne gerek var? Para kazandıracak mı? Yeni bir programlama dilini duyduğumuzda, hepimizin aklına gelen soru bu. Kitabın girişinde öncelikle bu soruyu yanıtlamaya çalışacağız.

Go (ya da başka bir deyişle: Golang), Google'ın büyük ve dağıtık uygulamalarında yaşadığı ihtiyaçtan yola çıkılarak geliştirilmiş bir programlama dilidir. Çok kabaca bir bakış ile, "performans odaklı, genel amaçlı, derlenen ve açık kaynaklı" bir dil olduğunu söyleyebiliriz.

Her programlama dili, ihtiyaçlardan dolayı ortaya çıkar. Hızlı gelişen bilişim teknolojilerine bağlı olarak sürekli yeni ihtiyaçlar ve yeni çözümler karşımıza çıkmaktadır. 2017 yılı için gündemdeki programlama eğilimlerine hızlıca bakıldığında gördüğümüz bazı kavramlar şöyle: büyük veri, nesnelerin interneti (IoT), veri analizi, veri madenciliği (ve diğer yapay zeka teknolojileri), dağıtık mimariler, NoSQL, sanallaştırma ve imaj yönetimi, bulut mimarileri (IaaS, PaaS, vb.), gömülü sistemler, paralel hesaplama, mobil programlama, sanal (artırılmış) gerçeklik, mikroservisler ve tabii ki performans.

CSS, MVC (MVT), ORM, JavaScript, Nesne Tabanlı Programlama, sürüm takip sistemleri, web çatıları, frontend/backend, vb. konular ise henüz gündemden düşmedi, artık tüm yazılımcıların bunları yalayıp yuttuğu kabul ediliyor. Gündemdeki konuların yukarıda sıralanmasında, "hepsinin öğrenilmesi gerektiği" veya "moral bozma" gibi bir amacımız yok tabii ki. Teknolojik gelişmeleri doğru yorumlayabilmek için bu ortamı hazırlayan ihtiyaçların iyi görülebilmesi gerekmektedir. **Bu kitabın hedefi;** Dünyada büyük ölçekte kullanım örnekleri olan, ülkemizde pek bilinmeyen ve yakın gelecekte gündemde önemli bir yeri olacağına inandığımız Go dili hakkında temel bilgiler verilmesinin yanı sıra, bu dilin farklı türde kullanım alanlarına dair uygulamalar ve verimli programlama örnekleri verilmesi olacaktır.

> **İPUCU** İnternet üzerinde, **Go** şeklinde arama yapıldığında, **Go Oyunu**'da dahil olmak üzere çok sayıda ilgimizi çekmeyen sonuç gelmektedir. Arama yaparken, **Go** yerine **Golang** sözcüğünü kullanmayı deneyebilirsiniz.

Bu Kitap Kime Göre?

Hiç programlama bilmiyorsanız, bu kitap size göre değil. Çünkü bu kitap programlama öğretmek amacı ile yazılmadı. Daha önce başka programlama dilleri ile çalıştıysanız, biraz olsun bunlarda hakimiyetiniz varsa, Go dilini merak ediyor ve öğrenmek istiyorsanız, bu kitap tam size göre. Kitabın içeriğinde; Go'nun ortaya çıkış amacından, temel kurallarına ve farklı uygulama alanlarına kadar, temel ve orta seviyede birçok konu bulunmaktadır. İleri seviye projeler geliştirmek isteyenler için kitap içerisinde referanslar verilmiştir.

Go Nasıl Bir Dildir?

Go'nun resmi web sitesinde ***https://golang.org/*** Go şöyle tanımlanmıştır: "**Go;** basit, kararlı ve verimli yazılım oluşturmayı kolay bir şekilde sağlayan açık kaynaklı bir programlama dilidir".

Go için sıklıkla "pragmatik (faydacı, işlevsel)" ifadesi kullanılmaktadır. Bunun sebebi gerçek dünya problemleri için tasarlanmış olmasıdır. Bilimsel çalışmalar veya farklı özel amaçlar Go'nun öncelikli odağı değildir. Go sadelik ve faydayı, programcılık geleneklerinden önde görür.

İçerisinde paralel (çok görevli) çalışma özellikleri barındıran, performans ve basitlik odaklı, genel amaçlı bir dildir. Çok farklı alanlarda Go ile uygulama geliştirmek mümkündür. Web sunucusu, sistem programlama, API geliştirme, kullanıcı arayüzleri, nesnelerin interneti, grafik uygulamaları, makine öğrenmesi, doğal (native) mobil uygulama geliştirme gibi örnekler bu konuda sayılabilir.

Go, **statik tipli** ve **derlenen** bir dildir. Kod içerisinde değişkenler kullanılmadan önce değişkenin türü belirtilmelidir. Bu açıdan bakıldığında; Java, C, C++, C#, F#, Haskell, Objective-C vb. diller ile aynı türdendir. Diğer sınıfta ise **dinamik tipli** diller vardır. Bu türden dillere örnek olarak JavaScript, Lua, PHP, Python, Ruby gibi diller verilebilir.

Go Dilinin Temel ve Önemli Özelliklerine Kısaca Değinelim

Pragmatik ve minimalist bir dildir. Fayda ve basitlik odaklı olduğundan daha önce bahsetmiştik. Go'da sadece gerekli ve işlevsel olan eşyaları yanınıza alarak verimli uygulama geliştirebilirsiniz. Yazım biçimi C diline benzer, öğrenmesi ve kod okuması kolaydır.

Statik tiplidir, gerekirse dinamik tip de kullanılabilir. Bir dilin statik veya dinamik tipli olması, dilin karakteristiğini belirleyen önemli bir farktır. Statik dillerde, derleme sırasında (compile-time) değişken tip kontrolü yapıldığından, çalışma

3

Temel Veri Tipleri ve Değişken Kullanımı

Bu Bölümde

Bu bölümde, diğer dillerden de aşina olduğumuz temel veri türlerini Go üzerinde ele alacağız. Sonunda küçük uygulamalar yazmaya başlıyoruz.

Statik tipli dillerde değişkenlerin (verilerin) tipi baştan belirlenir ve program boyunca tipi değiştirilemez. Go da statik tipli dildir, bu nedenle değişkenlerin tipi mutlaka belirtilmelidir. Go'da kullanılan temel değişkenler; diğer dillerden de alışık olduğumuz tamsayı, gerçek sayı, karakter, mantıksal gibi türlerdir. Aşağıdaki bölümlerde, Go dili içinde hazır olarak gelen değişken tipleri örneklerle açıklanmıştır.

Bir değişken tanımlama işlemi, basitçe, `var degişken_adı veri_türü` şeklinde yapılmaktadır. `var` ifadesi, **variable** (değişken) sözcüğünün kısaltmasıdır.

NOT C ve diğer bazı dillerde değişken tanımlarken, `veri_türü degişken_adı` (örneğin: `int i`) şeklinde yazılmaktadır. Go'da bunun tersi şekilde değişken tanımlanması, konuşma diline daha yakın olduğu için tercih edilmiştir.

TAMSAYI (INTEGER)

Ondalık bileşeni olmayan sayılardır. -176, -3, 0, 2, 1524 gibi sayılardır. Bilgisayarda ikilik sayı sistemi geçerli olduğundan, değişkenler de ikilik olarak kullanılır. Tamsayı değişkenler, kendileri için ayrılan bit sayısı kadar büyüklüğe sahip olabilirler. Örneğin; 4 bitlik bir pozitif (işaretsiz) tamsayı ile 24=16 tane sayı (0-15) ifade edilebilir. Negatif tamsayıların kullanılması durumunda, 1 bit de işaret için kullanılacaktır.

Go dilinde gelen tamsayı türleri; `int`, `int8`, `int16`, `int32`, `int64`, `uint`, `uint8`, `uint16`, `uint32`, `uint64`, `uintptr` şeklindedir. İfadelerin başındaki u harfi, `unsigned` (işaretsiz, pozitif) anlamındadır. 8, 16, 32 ve 64 ifadeleri de tamsayının kaç bitlik olduğunu belirtir.

Tamsayı türlerinde, bit sayısı belirtilmemiş olan `int`, `uint` ve `uintptr` türleri ise kullanılan sisteme bağlıdır. Genellikle 32 bitlik sistemlerde bu türler de 32 bit olur. Benzer şekilde, 64 bitlik sistemlerde de bu tamsayılar 64 bit olarak kullanılmaktadır.

İPUCU Go dilinde bir tamsayı değişkene ihtiyaç olduğunda, genellikle sadece `int` kullanılması yeterlidir.

İki tane tamsayı değişken tanımlayan ve bu değişkenlerin toplamını ekrana yazan örnek program kodu aşağıda verilmiştir:

```
// uygulama_3_1
package main
import "fmt"
func main() {
          var sayi1, sayi2 int
          sayi1 = 3
          sayi2 = sayi1 + 5
          fmt.Println("Sayıların toplamı:", sayi1+sayi2)
}
```

NOT Go dilinde **UTF-8** desteği doğrudan gelmektedir. Metinlerle ilgili işlemlerde Türkçe karakterler de (ğ, ş, ı, İ, ö, ü) dahil olmak üzere tüm UTF-8 karakterler sorunsuzca kullanılabilir.

Yukarıdaki program kodu çalıştırıldığında ekranda `Sayıların toplamı:` 11 ifadesi yazmaktadır. Görüldüğü gibi bir satırda aynı türde birden fazla değişken tanımlanabilmektedir. Değişken tanımlama konusunda farklı yazım örnekleri sonraki konularda yapılacaktır.

Kayan Noktalı Sayılar (Floating-Point)

Gerçek (reel) sayılar için kullanılır. Sayıların ondalık hanesi, nokta işareti (.) ile ayrılarak yazılır. 3.14, 54.11, -5.000002 gibi sayılar için kullanılabilir. Kayan noktalı sayıların bilgisayarda ikilik sistem kullanılarak temsil edilmesi karmaşık bir işlemdir ve bu kitabın kapsamı dışındadır.

DİKKAT Kayan noktalı sayı değişkenleri **tam değeri** ifade etmeyebilir. Sayının ondalık hanesi, değişken için ayrılan bellek alanına sığmazsa, düşük öncelikli basamağı kırpılarak belleğe yazılır. Hassasiyet, kullanılan bit sayısına bağlıdır.

Go dilinde, `float32` ve `float64` şeklinde iki tane kayan noktalı veri türü vardır. Adından da anlaşılacağı gibi bu veri türleri 32 ve 64 bit alanda depolanır. Hassasiyetin fazla olması isteniyorsa, 64 bit kullanılması tercih edilebilir. `Float` olarak tanımlanmış bir değişkenin değeri; `NaN` (Not A Number ~ sayısal olmayan değer) ve ±∞ şeklinde de olabilir. Bu tarz değerler, sıfıra bölme hatası gibi beklenmeyen durumlarda çıkabilmektedir.

6

Fonksiyonlar ve İşaretçiler

Bu Bölümde

Bu bölümde işlevsel bloklar halinde program geliştirmeyi sağlayan fonksiyonlar, büyük projelerde çok sayıda fonksiyonlarla çalışıldığında işimizi kolaylaştıracak bazı özellikler ve son olarak fonksiyonlar arasında veri aktarımını kolaylaştıran işaretçi değişkenler üzerinde durulacaktır.

FONKSİYON NEDİR? NASIL KULLANILIR?

Programlama dillerinde fonksiyon; veri girişi ve veri çıkışı yapabilen, belirli bir amaca yönelik olarak tanımlanmış işlemleri yapan kod bloğudur. Fonksiyon yerine **prosedür**, **yordam**, **alt rutin**, **alt program** gibi isimler de kullanılabilmektedir.

Tanımda yazılan **veri girişi ve veri çıkışı yapabilen** şeklindeki ifadeyi biraz daha açalım. Fonksiyonlar cevap olarak veri döndürebilir. Örneğin; bir fonksiyona **bana rasgele sayı üretir misin?** diyebiliriz. Eğer fonksiyon bu amaçla yazılmışsa, cevap olarak rasgele bir sayı döndürecektir.

Fonksiyon, bazı verileri alıp, bunları işleyip, geriye bir değer döndürmeden de çalışabilir. Mesela kendisine verilen iki değişkenin içeriklerini yer değiştiren bir fonksiyon yazılabilir.

Bir fonksiyon, hiçbir veri almadan ve geriye değer döndürmeden de çalışabilir. Ekrana belirli bir metni yazan bir fonksiyon olabilir. Fonksiyonun ekrana yazı yazması bir işlemdir, bunu geriye cevap döndürdüğü şeklinde yorumlamayınız.

Bazen de fonksiyonlar bir veya birden fazla veri alabilir, bir veya birden fazla cevap döndürebilir. Örneğin; kendisine verilen sayıların ortak çarpanlarını hesaplayıp geriye cevap olarak döndüren bir fonksiyon yazılabilir.

NOT Fonksiyonlar ihtiyaç oldukça tekrar tekrar çağırılabilir. Birden fazla yerde ihtiyaç olacağını düşündüğünüz işlemleri fonksiyon ile gerçekleştirebilirsiniz.

Fonksiyon kullanımının başka bir faydası, program kodlarının işlevsel bloklar halinde olmasını sağlamasıdır. Bu sayede hata bulma/düzeltme, geliştirme, projeyi genişletme ve sadeleştirme gibi işlemler daha kolay olacaktır.

Her Go programında en az bir fonksiyon olmak zorundadır. Önceki uygulamaların tamamında `func main(){…}` şeklinde tanımlanan ana fonksiyonu kullanmıştık. Bu bölümde `main` fonksiyonunun dışında başka fonksiyonlar da oluşturacağız. Fonksiyonların genelleştirilmiş yazım biçimi aşağıdaki gibidir:

```
func fonksiyon_ismi(giriş) cevap {
// komutlar
//…
return cevap
}
```

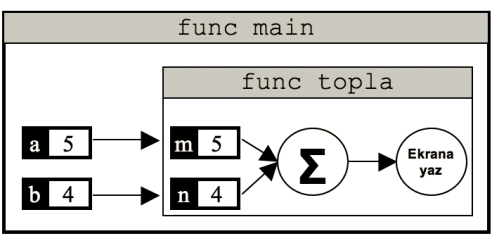
Şimdiye kadar gördüğümüz uygulamalarda `func main()` şeklinde tanımlanmış olan ana fonksiyonun giriş verisi almadığını anlayabiliyoruz, çünkü parantez içinde bir argüman bulunmamaktadır. Şimdi çok basit bir fonksiyon uygulaması yapalım. Bu uygulamada İki sayının toplamını ekrana yazan bir fonksiyon olsun:

```
// uygulama_6_1
package main

import "fmt"

func main() {
          a := 5
          b := 4
          topla(a, b)
}
// topla isimli fonksiyon aşağıda tanımlanıyor.
func topla(m, n int) {
          fmt.Println("Sayıların toplamı=", m+n)
}
```

Fonksiyonların sırası önemli değildir; ana fonksiyondan önce veya sonra yazılabilir. Yukarıdaki uygulamada `topla` ismindeki fonksiyon, iki tane tamsayı değer alabiliyor. Aldığı değerlerin toplamını ekrana yazdırıyor. Verilen örnekte dikkat edilmesi gereken nokta; fonksiyona gönderilen değişkenler ve bunların fonksiyon içinde farklı isimle işlenmesidir. Aşağıda bu yapı, grafik olarak gösterilmiştir.



Fonksiyon dışında `a` ve `b` değişkenleri içerisinde tutulan veriler, fonksiyona girerken `m` ve `n` değişkenlerine aktarılmıştır. `m` ve `n` değişkenleri sadece fonksiyon içinde tanımlıdır. Hatırlatalım, fonksiyon dışında bu değişkenler kullanılamazlar.

Hadi yukarıdaki programda kullandığımız topla fonksiyonunu biraz daha geliştirelim. Bu sefer ekrana yazmak yerine, geriye değer döndürsün. Cevap olarak gelen değeri bir değişkene atalım ve istediğimiz gibi kullanalım. Aşağıda, bu işleri yapan program kodu verilmiştir:

10

Web Programlama

Bu Bölümde

Bu bölümde; Go içerisinde gelen temel web programlama araçlarının yanı sıra, veritabanı kullanımı ve farklı ekipler tarafından geliştirilen güncel web çatıları gibi konular ele alınmıştır.

Bu bölümdeki uygulamalarda biz de hazır gelen bileşenlerle web programcılığı yapacağız. Bölümün sonunda çatı kullanımı konusunda da kısa bilgi verilecektir.

Statik Web Sunucusu

Statik web sayfaları, bir motor tarafından üretilmezler. Sayfa her çağrıldığında aynı cevap alınır. Herhangi bir yere veya değişkene bağlı değildir. Örneğin; stil dosyası olan CSS dosyaları, resimler, videolar, HTML dosyaları gibi dosyalar statik içeriklerdir. Şimdi bir uygulama yapacağız ve bu uygulamada aşağıdaki dosya/klasör yapısını oluşturacağız:

```
├── statik
│   ├── main.css
│   ├── gopher.png
│   └── index.html
└── main.go
```

Yukarıda `main.go` ismi ile belirtilen dosya, çalıştıracağımız Go dosyası. Bu dosya çalıştırılınca, bilgisayarda TCP 8000 portunu dinlemeye başlayacak. Web sunucusuna, / şeklinde bir web isteği gelirse, bu isteği/statik klasörüne yönlendirecek. Önceki bölümde net/http paketi ile ilgili verilen örnekte bu kadarı yapılmıştı aslında. Önce ana klasördeki `main.go` dosyasını görelim:

```
// uygulama_10_1
// main.go
package main

import (
          "log"
          "net/http"
)

func main() {
          http.Handle("/", http.FileServer(http.Dir("statik")))
          log.Println("TCP 8000 portunu dinliyorum...")
          http.ListenAndServe(":8000", nil)
}
```

Yukarıdaki program çalıştırıldığında, TCP 8000 portunu açarken bir sorun yaşamazsa, ekranda 2016/11/20 18:47:16 TCP 8000 portunu dinliyorum... şeklinde mesaj yazmakta ve gelen HTTP isteklerine cevap vermektedir.

İPUCU Go ile hazır gelen log paketi, standart çıktı aygıtına olay bilgilerini yazmak için kullanılır. Her bir mesaj ile birlikte zaman bilgisini de otomatik olarak yazar.

http.Handle ile başlayan satırda, / şeklinde gelen klasör isteklerinin /statik web klasörüne yönlendirilmesi işlemi yapılmaktadır. statik klasöründe index.html isimli dosya, bu klasör içerisinde web üzerinden sunulacak olan dosyadır. Bu dosyanın web üzerinden çağırılması için, tarayıcıda http://localhost:8000/index.html adresi kullanılmalıdır. *index.html* ismi uzun zamandır web sunucularında varsayılan olarak kullanılmakta olduğundan, Go dilinde de eğer dosya ismi kısmı yazılmazsa yine aynı dosya getirilecektir. Yani aynı adrese http://localhost:8000/ şeklinde de erişebiliriz. *index.html* dosyasının içeriği aşağıda verilmiştir:

```
<!-- index.html -->
<html>
<head>
    <meta charset="utf-8">
    <title>Statik sayfa</title>
    <link rel="stylesheet" href="/main.css">
</head>
<body>
    <h1>Merhaba!</h1><h2>Ben, Go içinden çağırılan bir statik sayfayım.</h2>
    <p>CSS dosyaları, resimler, sabit HTML dosyaları gibi dosyalar, statik içeriktir.</p>
    <img src="/gopher.png" width="300">
</body>
</html>
```

Statik klasöründeki main.css ve gopher.png dosyaları, index.html içerisinden çağırılmaktadır. main.css dosyası, stilleri belirtmek için kullanılmıştır. Bu basit uygulamada özellikle kullanılmış olmasının sebebi, bir resim dosyası ile beraber statik içeriklere örnek olmasıdır. Aşağıda main.css içeriği verilmiştir:

```
body {
    border: 2px solid red;
    border-radius: 20px;
    border-width: 5px;
    text-align: center;
```

```
    width: 400px;
    margin: auto;
}
```

Tarayıcı üzerinden `http://localhost:8000/` adresine girildiğinde, bütün dosyalar görevini yapmakta ve tarayıcı penceresinde şu ekran görülmektedir:

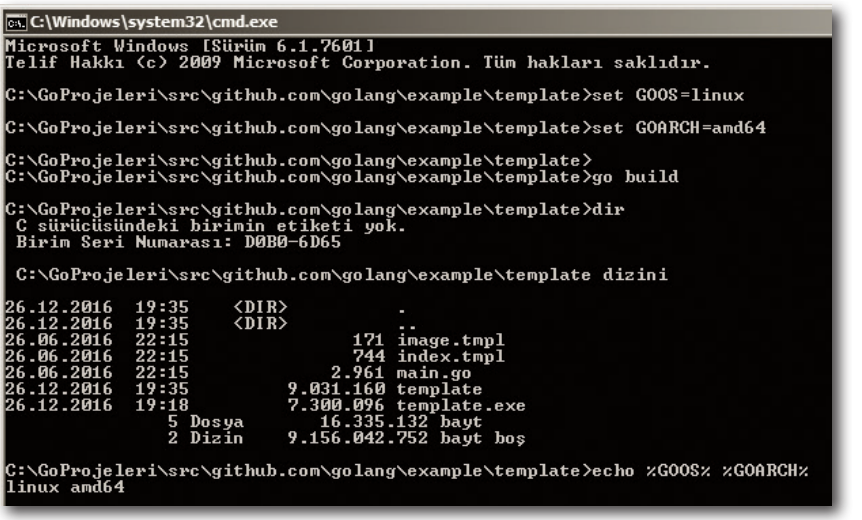
Statik içerik kullanım örneği bu kadar. Ancak sadece statik içerik kullanılacaksa, zaten HTML'den başka bir dile ihtiyaç bulunmamaktadır. Web programlama dillerinin asıl önemi dinamik içeriklerde ortaya çıkmaktadır. Sonraki başlıklarda dinamik web sayfaları konusu ele alınacaktır.

HTML Şablon (Template) Kullanımı

HTML şablonları, web sayfası görünümünü (tasarımını) belirten şablonlardır. Karmaşık işleri yapan program kodlarını (fonksiyonlarını) web tasarımından ayırmak için kullanılır. Bu sayede farklı uzmanlık alanları gerektiren işler birbirinden ayrılmış olur. Bir kişi sadece görsel tasarım ile uğraşırken, başka bir kişi de görsel tasarımda kullanılacak olan verilerin hazırlanması ve sunulması ile uğraşabilir.

Go dilinde HTML şablon kullanımı için `html/template` isimli bir paket gelmektedir. Daha önce kullandığımız `net/http` paketi ile beraber, bu ikili oldukça güzel olanaklar sunmaktadır. Bu bölümde yapacağımız örnek uygulamada iki tane dosya oluşturacağız. Bir tanesinde, klasik işlerimizi yapan Go kodlarını yazacağız. Diğer dosyada ise HTML kodları bulunacak. Web sayfasında görüntülemek istediğimiz verilerin değişkenlerini HTML kodlarının aralarında kullanacağız. Özetle şöyle bir şey olacak:

Benzer şekilde; Windows bilgisayar üzerinde de çevresel değişkenler kullanılabilir. Windows'ta bir çevresel değişkeni komut satırında değiştirmek için, SET komutu kullanılabilir. Ya da istenirse grafik arayüz üzerinden de bu değişiklik yapılabilir. Aşağıdaki ekran görüntüsünde, set GOOS=linux ve set GOARCH=amd64 şeklinde ilgili değişkenler belirlenmiştir. Sonrasında go build komutu ile derleme yapılmış ve 64 bitlik Linux üzerinde çalıştırılabilecek olan bir dosya (resimde görülen template isimli dosya) oluşmuştur.



Windows üzerinde echo %GOOS% ve %GOARCH% komutu ile ilgili değişkenlerin değeri görülebilir.

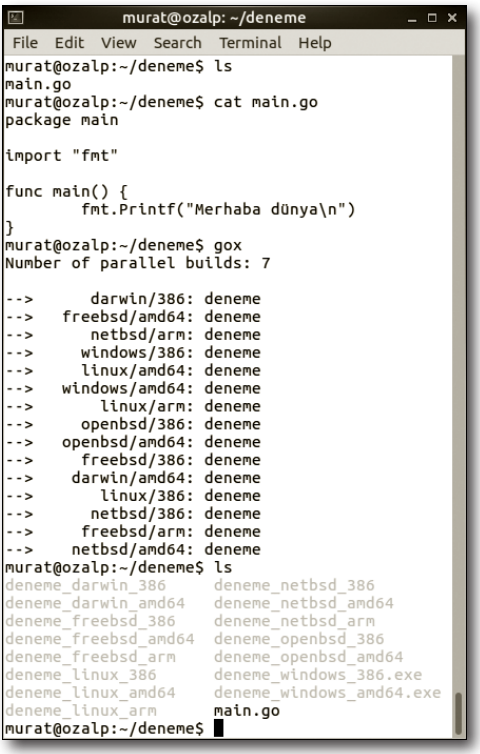
> **İPUCU** Eğer farklı platformlar için derleme işlemi sıkça yapılan bir işlem ise bunun için yazılmış olan kütüphanelerden faydalanılabilir.

Farklı platformlar için derleme işlemini kolaylaştıran bir uygulama ile ilgili örnek yapalım. Kullanımı çok basit olan **gox** paketi ***github.com/mitchellh/gox*** adresinden aşağıdaki gibi yüklenebilir:

```
go get github.com/mitchellh/gox
```

gox uygulaması derleme yapılan bilgisayarın işlemcisindeki çekirdek sayısına göre, paralel olarak derleme işlemi yapmaktadır. İstenirse desteklenen tüm platformlar için kolayca derleme yapılabileceği gibi, bazı platformlar seçilerek te derleme yapılabilir. Ekran görüntüsünde gox kullanımı verilmiştir. Go proje-

sinin olduğu klasörde sadece gox yazılarak, desteklenen tüm platformlar için derlenmiş dosyalar (*binary*) oluşturulmuştur. gox çalışmaya başladığında, paralel derleme işlemini 7 süreç ile yaptığını belirtmiştir.



Yazılım Testleri

Yazılım projelerinde; analiz yapılması, kodların yazılması, derlenmesi, test edilmesi, onaya sunulması, yayınlanması, entegrasyon yapılması, geri beslemelerin alınması, güncelleştirme ve düzeltmelerin yapılması vb. şeklinde devam eden aşamalar bulunmaktadır. Bu aşamalar genellikle çevrim şekilde birbirini tekrarlar, çünkü yazılım projeleri canlıdır. Bu çevrime, **yazılım yaşam döngüsü** ismi verilir. Yaşam döngüsünün aşamaları, proje sahibinin yöntemlerine göre küçük farklılıklar gösterse de genel hatlarıyla her projede benzer işlemler yapılır. Bazı projelerde, her aşamayı farklı kişiler yaparken, bazen tek kişinin (Süpermen) tüm aşamaları üstlendiği de olabilir. Kurumdan kuruma, projeden projeye farklılık gösteren bu sürecin, yazılımcıları özellikle ilgilendiren bazı aşamalarını burada örneklendirmeye çalışacağız.