Pemrograman Go

Dr. Bambang Purnomosidi D. P. <bambangpdp@gmail.com>

Version 0.0.1-rev-2021-03-21 12:56:44 +0700

Daftar Isi

Preface

Buku ini merupakan buku bebas tentang bahasa pemrograman Go. Go merupakah bahasa pemrograman yang dirancang dan pertama kali diimplementasikan oleh Robert Griesemer, Rob Pike, dan Ken Thompson di Google. Go banyak digunakan pada sistem terdistribusi dan sistem berbasis Cloud. Buku ini dibuat dengan sponsor dari Zimera Systems dan mempunyai lisensi **Creative Commons Attribution-ShareAlike 4.0 International**.

- Lisensi dalam Bahasa Indonesia - https://creativecommons.org/licenses/by-sa/4.0/deed.id.
- Lisensi dalam Bahasa Inggris - https://creativecommons.org/licenses/by-sa/4.0/deed.en.

Secara umum, penggunaan lisensi ini mempunyai implikasi bahwa pengguna materi:

1. Harus memberikan atribusi ke penulis dan sponsor untuk penulisan materi ini.
2. Boleh menggunakan produk yang ada disini untuk keperluan apapun jika point 1 di atas terpenuhi.
3. Boleh membuat produk derivatif dari produk yang ada disini sepanjang perubahan-perubahan yang dilakukan diberitahukan ke kami dan di-share dengan menggunakan lisensi yang sama.

Atribusi

Dr. Bambang Purnomosidi D. P.
Zimera Systems
Dusun Medelan, Umbulmartani, Ngemplak
Sleman, DIY
https://www.google.com/maps/place/Zimera+Systems/@-7.6975303,110.43921,17z/data=!3m1!4b1!4m5!3m4!1s0x2e7a5d7cc40e8871:0x2d44da15f0b37

81e!8m2!3d-7.6975303!4d110.4413987
E-mail: zimera-systems@gmail.com

Pembuatan buku ini merupakan hasil pekerjaan kolektif baik secara langsung maupun tidak langsung. Penulis serta kontributor mengucapkan terima kasih untuk Zimera Systems yang telah memberikan **sponsorship** selama penulisan buku ini.

Bagian I: Go dan Peranti Pengembangan Go

Bagian ini membahas tentang pengenalan Go secara umum dan bagaimana menyiapkan peranti pengembangan jika ingin membuat program meggunakan Go.

Bab 1. Pengenalan Go

1.1. Apa itu Go?

Go adalah nama bahasa pemrograman sekaligus nama implementasi dalam bentuk kompilator (`compiler`). Untuk pembahasan berikutnya, istilah `Go` akan mengacu juga pada spesifikasi bahasa pemrograman serta peranti pengembangannya. Nama yang benar adalah `Go`, bukan `Golang` atau `golang`. Istilah `Golang` atau `golang` muncul karena nama domain `go.org` tidak tersedia saat itu dan mencari sesuatu melalui Google atau mesin pencari lainnya menggunakan kata kunci `Go` tidak menghasilkan hasil yang baik. Dengan demikian, untuk penyebutan di **hashtag** biasanya digunakan `#golang` sehingga mesin pencari bisa mengindeks dan memberikan hasil yang baik. Lihat FAQ tentang Go di https://golang.org/doc/faq#go_or_golang untuk informasi lebih lanjut.

1.2. Lisensi Go

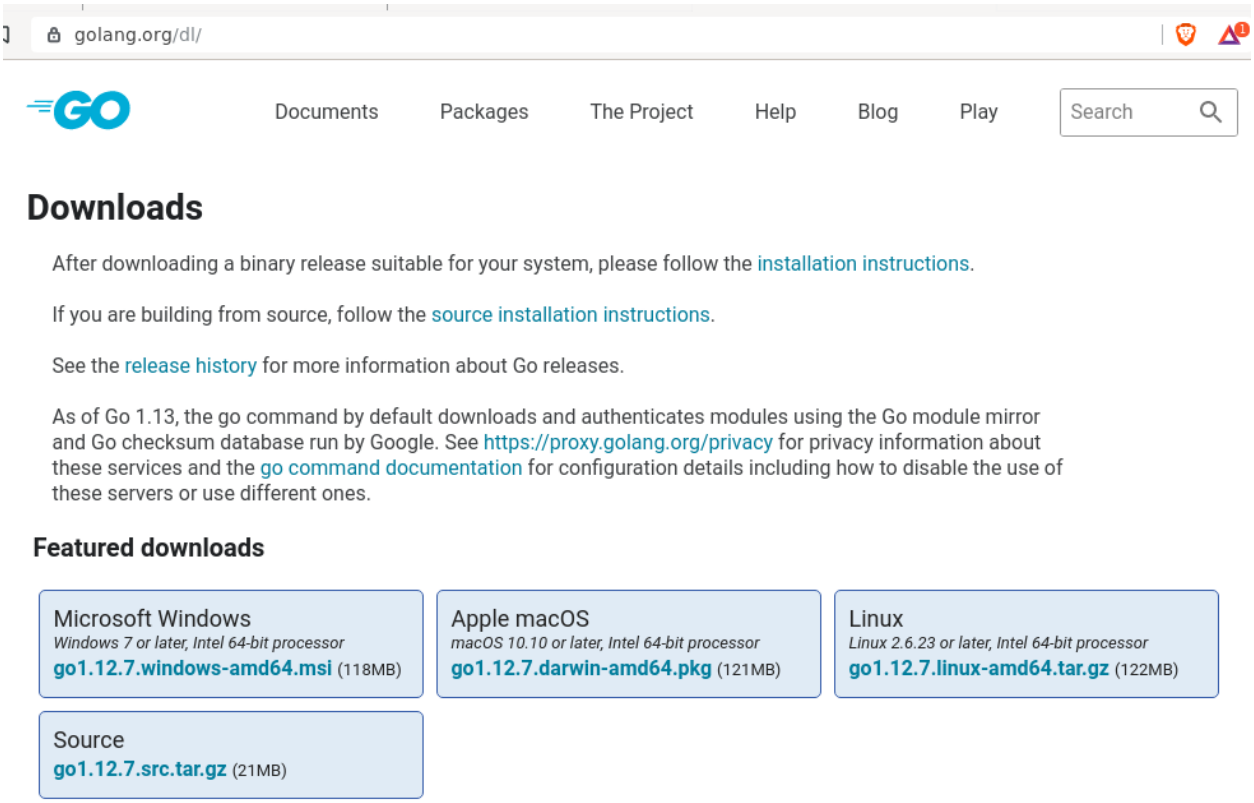
Go didistribusikan dengan menggunakan lisensi modifikasi dari BSD. Lisensi lengkap dari Go bisa diakses di Lisensi Go. Secara umum, penggunaaan lisensi ini mempunyai implikasi sebagai berikut:

- boleh digunakan untuk keperluan komersial maupun non-komersial tanpa batasan
- boleh memodifikasi sesuai keperluan
- boleh mendistribusikan
- boleh memberikan sublisensi ke pihak lain
- boleh memberikan garansi
- tidak boleh menggunakan merk dagang Go
- tanpa jaminan dan jika terjadi kerusakan terkait penggunaan software ini maka pemberi lisensi tidak bisa dituntut
- jika mendistribusikan harus mengikutsertakan pemberitahuan hak cipta.

1.3. Instalasi Go

Go tersedia pada berbagai platform. Proyek Go sendiri secara resmi mendukung platform Linux, FreeBSD, MacOSX, dan Windows. Dukungan tersebut merupakan dukungan resmi dan

distribusi `binary executable` dari berbagai platform tersebut tersedia di repository download Go seperti pada gambar berikut:



Dengan dukungan tersebut, Proyek Go akan menerima laporan `bugs` terkait dengan distribusi pada berbagai platform tersebut. Meski demikian, bukan berarti platform-platform lain tidak bisa menggunakan Go karena distribusi dalam bentuk kode sumber tersedia dan telah berhasil dikompilasi ke berbagai platform: NetBSD, OpenBSD, DragonFlyBSD, dan lain-lain. Informasi mengenai platform-platform yang mungkin bisa digunakan oleh Go bisa diperoleh di wiki.

1.3.1. Download dan Install Go

Download dan instalasi Go pada tulisan ini adalah download dan instalasi untuk lebih dari satu versi Go dan masing-masing menggunakan workspace sendiri-sendiri. Hal ini disebabkan karena seringkali software yang dibangun ditargetkan untuk lebih dari satu versi, misalnya Go 1.11 ke atas (Go 1.11.x, 1.12.x, dan 1.13.x). Kondisi ini menjadi tidak sederhana karena penulis tidak ingin mencampuraduk kode sumber yang dibuat menggunakan masing-masing versi. Go sendiri menyarankan untuk menggunakan satu workspace untuk semua proyek Go yang kita buat. Satu workspace saja tidak masalah jika hanya mentargetkan satu versi. Di bagian ini penulis akan menjelaskan konfigurasi yang penulis gunakan untuk menangani masalah tersebut.

Lokasi instalasi Go

Go akan diinstall di direktori $HOME/software/go-dev-tools/goVERSI

VERSI = versi dari Go yang akan diinstall, misalnya go1.12.7

Lokasi instalasi tersebut digunakan penulis karena penulis mempunyai lebih dari 1 versi Go, jika nanti ada versi lainnya, versi lain tersebut akan di-install (misal versi 1.13.0) di $HOME/software/go-dev-tools/go1.13.0

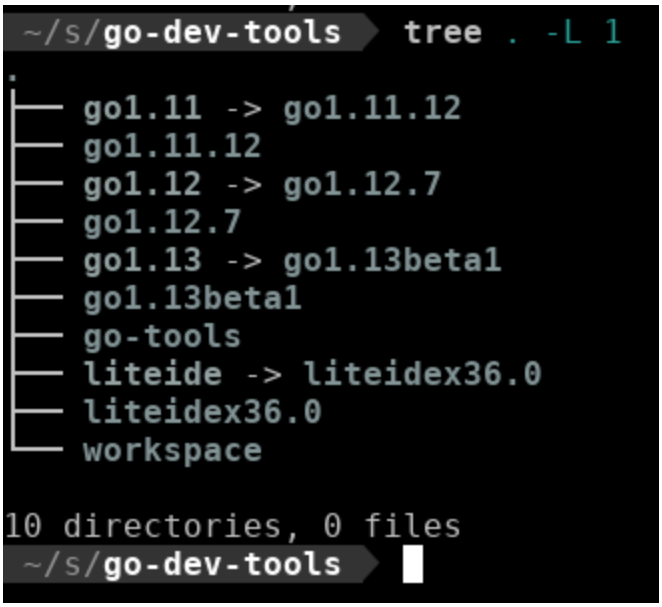
Meski mendukung banyak platform, di buku ini hanya akan dibahas penggunaan Go di platform Linux. Pada dasarnya peranti pengembang yang disediakan sama. Silahkan menyesuaikan dengan platform yang anda gunakan. Untuk instalasi berikut ini, ambil distribusi yang sesuai dengan platform di komputer anda. Untuk pembahasan ini, digunakan `go1.12.7.linux-amd64.tar.gz`. Setelah itu, ikuti langkah-langkah berikut:

```
$ ls -la
total 475520
drwxr-xr-x   4 bpdp bpdp      4096 Jul 21 08:16 ./
drwxr-xr-x  88 bpdp bpdp      4096 Jul 12 14:17 ../
-rw-r--r--   1 bpdp bpdp 127959471 Jul 19 04:41 go1.12.7.linux-
amd64.tar.gz
...
...
$ mkdir -p $HOME/software/go-dev-tools
$ cd $HOME/software/go-dev-tools
$ tar -xvf ~/master/go/go1.12.7.linux-amd64.tar.gz
$ mv go go1.12.7
```

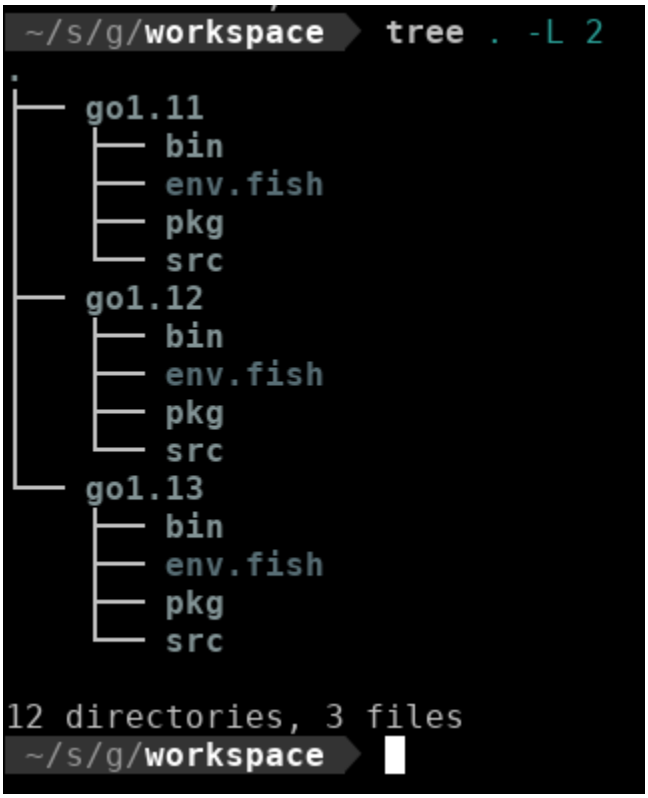
Setelah menjalankan langkah-langkah di atas, Go sudah terinstall di direktori `$HOME/software/go-dev-tools/go1.12.7`

1.3.2. Konfigurasi Variabel Lingkungan Sistem Operasi, Compiler Go, dan Workspace

Untuk konfigurasi kompiler, ada tiga langkah yang perlu dilakukan: download, ekstrak pada lokasi tertentu, dan terakhir setting environment variables. Pada konfigurasi ini, compiler dan workspace berada pada `$HOME/software/go-dev-tools/`. Lokasi ini selanjutnya akan kita sebut dengan GODEVTOOLS_HOME. Setelah download dan install compiler Go seperti langkah di atas, buat struktur direktori sebagai berikut (untuk go1.12.7 sudah dibuat dengan cara di atas):



Semua versi Go ada di $GODEVTOOLS_HOME. Direktori workspace digunakan untuk menyimpan kode sumber yang kita buat seesuai dengan versi Go yang kita targetkan. Untuk setiap direktori di workspace, buat struktur dan 1 file env.sh sebagai berikut:



Isi dari file env.sh menyesuaikan shell yang digunakan:

Bash

```
export GOPATH=`pwd`
export PATH=$PATH:$GOPATH/bin
```

Fish

```
set -x GOPATH (pwd)
set -x PATH $PATH $GOPATH/bin
```

Go menggunakan beberapa variabel lingkungan sistem operasi. Supaya berfungsi dengan baik, tetapkan nilai-nilai variabel lingkungan tersebut di file inisialisasi shell. Jika menggunakan **Fish**, maka inisialisasi tersebut ada di `$HOME/.config/fish/config.fish`. Jika menggunakan **Bash**, maka inisialisasi tersebut diletakkan di `$HOME/.bashrc`). Meski bisa diletakkan pada file tersebut, penulis menyarankan untuk meletakkan pada suatu file text biasa dan kemudian di-**source**. Pada bagian ini, penulis akan meletakkan di file `%HOME/env/fish/go/go1.12`.

```
set GODEVTOOLS_HOME /home/bpdp/software/go-dev-tools

set GO_HOME $GODEVTOOLS_HOME/go1.12
set LITEIDE_HOME $GODEVTOOLS_HOME/liteide
set GOTOOLS $GODEVTOOLS_HOME/go-tools

set -x GOROOT $GO_HOME
set -x GOOS linux
set -x GOARCH amd64
set -x GOHOSTOS linux
set -x GOHOSTARCH amd64

alias godev='cd $GODEVTOOLS_HOME/workspace/go1.12'
alias godevtools='cd $GOTOOLS'

set -x PATH $PATH $GO_HOME/bin $LITEIDE_HOME/bin $GOTOOLS/bin
```

Jika menggunakan **Bash**:

```
GODEVTOOLS_HOME=/home/bpdp/software/go-dev-tools

GO_HOME=$GODEVTOOLS_HOME/go/go1.12.7
LITEIDE_HOME=$GODEVTOOLS_HOME/liteide
GOTOOLS=$GODEVTOOLS_HOME/go-tools

export GOROOT=$GO_HOME
export GOOS=linux
export GOARCH=amd64
export GOHOSTOS=linux
export GOHOSTARCH=amd64

export PATH=$PATH:$GO_HOME/bin:$LITEIDE_HOME/bin:$GOTOOLS:
$GO3RDPARTYTOOLS/bin

alias godev='cd $GODEVTOOLS_HOME/workspace/go1.12'
alias godevtools='cd $GOTOOLS'
```

Dengan memasukkan beberapa variabel lingkungan tersebut ke file, saat kita ingin menggunakan Go, tinggal di - **source** sebagai berikut:

```
$ source ~/env/fish/go/go1.12.7
```

Setelah itu, Go bisa digunakan. Untuk melihat hasil, eksekusi perintah `go env`, hasilnya seharusnya adalah sebagai berikut:

```
$ go env
GOARCH="amd64"
GOBIN=""
GOCACHE="/home/bpdp/.cache/go-build"
GOEXE=""
GOFLAGS=""
GOHOSTARCH="amd64"
GOHOSTOS="linux"
GOOS="linux"
GOPATH="/home/bpdp/go"
GOPROXY=""
GORACE=""
GOROOT="/home/bpdp/software/go-dev-tools/go1.12"
GOTMPDIR=""
GOTOOLDIR="/home/bpdp/software/go-dev-tools/go1.12/pkg/tool/linux_amd64"
GCCGO="gccgo"
CC="gcc"
CXX="g++"
CGO_ENABLED="1"
GOMOD=""
CGO_CFLAGS="-g -O2"
CGO_CPPFLAGS=""
CGO_CXXFLAGS="-g -O2"
CGO_FFLAGS="-g -O2"
CGO_LDFLAGS="-g -O2"
PKG_CONFIG="pkg-config"
GOGCCFLAGS="-fPIC -m64 -pthread -fmessage-length=0 -fdebug-prefix
-map=/tmp/go-build584380045=/tmp/go-build -gno-record-gcc-switches"
$
```

Variabel $GOPATH seharusnya menunjuk ke workspace, baru akan berisi nilai yang benar (bukan $HOME/go) jika sudah men-**source** file `env.sh` di workspace.

Saat bekerja menggunakan Go, pada dasarnya kita akan menemukan berbagai macam proyek yang bisa dikategorikan menjadi 2 berdasarkan output dari proyek tersebut:

1. **Ready-to-use application**: aplikasi yang siap dipakai, biasanya didistribusikan dalam bentuk **binary executable(s)** atau kode sumber seperti nsq, Hugo, dan lain-lain.
2. Pustaka / **library** maupun aplikasi yang kita kembangkan sendiri.

Untuk dua kategori ini, ada dua perlakuan.

Ready-to-use application

Untuk kategori ini, siapkan lokasi khusus di media penyimpan untuk menyimpan hasil binary executable, setelah itu set `PATH`, `GOPATH` dan `go get -u -v <repo-url>`. Berikut adalah setting

pada komputer penulis:

```
~/s/g/go-tools  tree . -L 1
.
├── bin
├── env.fish
├── go-pkg-needed.sh
├── pkg
└── src

3 directories, 2 files
~/s/g/go-tools  cat go-pkg-needed.sh
#!/usr/bin/fish
#go get -u -v github.com/nsf/gocode
# diganti:
go get -u -v github.com/stamblerre/gocode
go get -u -v github.com/rogpeppe/godef
go get -u -v golang.org/x/lint/golint
go get -u -v github.com/lukehoban/go-outline
go get -u -v github.com/sqs/goreturns
go get -u -v golang.org/x/tools/...
go get -u -v github.com/uudashr/gopkgs
go get -u -v github.com/newhook/go-symbols
go get -u -v github.com/go-delve/delve/cmd/dlv
go get -u -v github.com/pointlander/peg
go get -u -v github.com/songgao/colorgo
go get -u -v github.com/motemen/gore
go get -u -v github.com/onsi/ginkgo/ginkgo
go get -u -v github.com/onsi/gomega/...
go get -u -v github.com/smartystreets/goconvey
go get -u -v github.com/blynn/nex
go get -u -v github.com/zmb3/gogetdoc
~/s/g/go-tools
```

Isi dari file `go-pkg-needed.sh` adalah sebagai berikut, anda bisa menambah atau mengurangi sesuai kebutuhan:

```
#!/usr/bin/fish

# ganti di atas dengan #!/usr/bin/bash jika anda menggunakan Bash
go get -u -v github.com/stamblerre/gocode
go get -u -v github.com/rogpeppe/godef
go get -u -v golang.org/x/lint/golint
go get -u -v github.com/lukehoban/go-outline
go get -u -v github.com/sqs/goreturns
go get -u -v golang.org/x/tools/...
go get -u -v github.com/uudashr/gopkgs
go get -u -v github.com/newhook/go-symbols
go get -u -v github.com/go-delve/delve/cmd/dlv
go get -u -v github.com/pointlander/peg
go get -u -v github.com/songgao/colorgo
go get -u -v github.com/motemen/gore
go get -u -v github.com/onsi/ginkgo/ginkgo
go get -u -v github.com/onsi/gomega/...
go get -u -v github.com/smartystreets/goconvey
go get -u -v github.com/blynn/nex
go get -u -v github.com/zmb3/gogetdoc
go get -u -v golang.org/x/tools/gopls
```

Dengan konfigurasi seperti itu, kerjakan berikut ini untuk install:

```
$ source env/fish/go/go1.12.7
$ godevtools
$ source env.sh
$ ./go-pkg-needed.sh
```

Perintah `source env.sh` di atas berguna antara lain untuk menetapkan `$GOPATH` ke `$GOTOOLS`. Setelah proses sebentar, hasil **binary executables** akan diletakkan pada $GOTOOLS/bin dan bisa kita jalankan langsung.

jangan meletakkan paket-paket **executables** ini jika $GOPATH belum menunjukkan nilai yang benar karena nanti akan tercampur dengan **binary executables** dari distribusi Go.

Pustaka / library maupun aplikasi yang kita kembangkan sendiri

Untuk keperluan ini biasanya kita menggunakan `modules` yang mulai ada pada versi Go 1.11 dan akan stabil pada versi 1.13. Modules ini akan kita bahas tersendiri.

1.3.3. Menguji Instalasi Go

Kode sumber Go yang kita buat bisa dijalankan / dieksekusi tanpa harus dikompilasi (jadi seperti script Python atau Ruby) atau bisa juga dikompilasi lebih dulu untuk menghasilkan `binary executable`. Selain menghasilkan `binary executable`, sebenarnya ada paket pustaka yang dimaksudkan untuk digunakan dalam program (disebut sebagai `package`). Package akan dibahas lebih lanjut pada bab-bab berikutnya.

Untuk menguji, buat program sederhana seperti listing `hello.go`. Setelah itu, gunakan `go run namafile.go` untuk menjalankan secara langsung atau dikompilasi lebih dulu dengan `go build namafile.go`.

```
// hello.go
package main

import "fmt"

func main() {
    fmt.Printf("hello, world\n")
}
```

Berikut ini adalah langkah-langkah untuk mengeksekusi `hello.go`:

```
$ go run hello.go
hello, world
$ go build hello.go
$ ls -la
total 1980
drwxr-xr-x 2 bpdp bpdp    4096 Jul 21 10:41 ./
drwxr-xr-x 3 bpdp bpdp    4096 Jul 21 10:40 ../
-rwxr-xr-x 1 bpdp bpdp 2014135 Jul 21 10:41 hello*
-rw-r--r-- 1 bpdp bpdp      86 Jul 21 10:40 hello.go
$ file hello
hello: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), statically
linked, Go
BuildID=-WQRW-exSunj5kUQwAX9/zYf98wtRiMVNHHsFRqn-/1wN0h--
29c_4cVsSKleo/4LgNCTrEswXoVuMCJgkH, not
stripped
$ strip hello
$ ls -la
total 1400
drwxr-xr-x 2 bpdp bpdp    4096 Jul 21 10:42 ./
drwxr-xr-x 3 bpdp bpdp    4096 Jul 21 10:40 ../
-rwxr-xr-x 1 bpdp bpdp 1420104 Jul 21 10:42 hello*
-rw-r--r-- 1 bpdp bpdp      86 Jul 21 10:40 hello.go
$ ./hello
hello, world
$ file hello
hello: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), statically
linked, Go
BuildID=-WQRW-exSunj5kUQwAX9/zYf98wtRiMVNHHsFRqn-/1wN0h--
29c_4cVsSKleo/4LgNCTrEswXoVuMCJgkH,
stripped
$
```

1.4. Memahami Lingkungan Peranti Pengembangan Go

Saat menginstall Go, kita akan memperoleh 3 buah file `binary executable`:

```
$ pwd
/home/bpdp/software/go-dev-tools/go1.12.7/bin
$ ls -la
total 34744
drwxr-xr-x  2 bpdp bpdp     4096 Jul  9 04:32 ./
drwxr-xr-x 10 bpdp bpdp     4096 Jul  9 04:29 ../
-rwxr-xr-x  1 bpdp bpdp 14617729 Jul  9 04:31 go*
-rwxr-xr-x  1 bpdp bpdp 17422226 Jul  9 04:32 godoc*
-rwxr-xr-x  1 bpdp bpdp  3525802 Jul  9 04:31 gofmt*
$
```

Penjelasan untuk masing-masing akan diuraikan di sub-sub bab berikut.

1.4.1. go

`go` merupakan peranti untuk mengelola kode sumber Go yang kita buat. Beberapa argumen dari `go` adalah:

```
$ go version
go version go1.12.7 linux/amd64
$ go
Go is a tool for managing Go source code.

Usage:

    go <command> [arguments]

The commands are:

    bug         start a bug report
    build       compile packages and dependencies
    clean       remove object files and cached files
    doc         show documentation for package or symbol
    env         print Go environment information
    fix         update packages to use new APIs
    fmt         gofmt (reformat) package sources
    generate    generate Go files by processing source
    get         download and install packages and dependencies
    install     compile and install packages and dependencies
    list        list packages or modules
    mod         module maintenance
    run         compile and run Go program
    test        test packages
    tool        run specified go tool
    version     print Go version
    vet         report likely mistakes in packages

Use "go help <command>" for more information about a command.

Additional help topics:

    buildmode   build modes
    c           calling between Go and C
    cache       build and test caching
    environment environment variables
    filetype    file types
    go.mod      the go.mod file
    gopath      GOPATH environment variable
    gopath-get  legacy GOPATH go get
    goproxy     module proxy protocol
    importpath  import path syntax
    modules     modules, module versions, and more
    module-get  module-aware go get
    packages    package lists and patterns
    testflag    testing flags
    testfunc    testing functions

Use "go help <topic>" for more information about that topic.
$
```

1.4.2. godoc

godoc merupakan peranti untuk menampilkan dokumentasi paket pustaka standar Go atau menampilkan server untuk dokumentasi Go (mirip seperti yang terdapat pada website dokumentasi Go.

```
$ godoc --help
usage: godoc -http=localhost:6060
  -analysis string
        comma-separated list of analyses to perform (supported: type,
pointer). See http://golang.org/lib/godoc/analysis/help.html
  -goroot string
        Go root directory (default "/home/bpdp/software/go-dev-
tools/go1.12")
  -http string
        HTTP service address (default "localhost:6060")
  -index
        enable search index
  -index_files string
        glob pattern specifying index files; if not empty, the index is
read from these files in sorted order
  -index_interval duration
        interval of indexing; 0 for default (5m), negative to only index
once at startup
  -index_throttle float
        index throttle value; 0.0 = no time allocated, 1.0 = full throttle
(default 0.75)
  -links
        link identifiers to their declarations (default true)
  -maxresults int
        maximum number of full text search results shown (default 10000)
  -notes string
        regular expression matching note markers to show (default "BUG")
  -play
        enable playground
  -templates string
        load templates/JS/CSS from disk in this directory
  -timestamps
        show timestamps with directory listings
  -url string
        print HTML for named URL
  -v    verbose mode
  -write_index
        write index to a file; the file name must be specified with
-index_files
  -zip string
        zip file providing the file system to serve; disabled if empty
$
```

1.4.3. gofmt

`gofmt` merupakan peranti untuk mem-format kode sumber dalam bahasa pemrograman Go.

```
$ gofmt --help
usage: gofmt [flags] [path ...]
  -cpuprofile string
        write cpu profile to this file
  -d    display diffs instead of rewriting files
  -e    report all errors (not just the first 10 on different lines)
  -l    list files whose formatting differs from gofmt's
  -r string
        rewrite rule (e.g., 'a[b:len(a)] -> a[b:]')
  -s    simplify code
  -w    write result to (source) file instead of stdout
$
```

Untuk melihat bagaimana `gofmt` bisa digunakan untuk membantu memformat kode sumber, buat kode sumber sederhana berikut ini:

```
// hello-unformatted.go
package main
import "fmt"
func main() {
    fmt.Printf("halo\n") // menampilkan tulisan
       fmt.Printf("dunia")  // ini tulisan baris kedua
}
```

Format file kode sumber di atas sebagai berikut:

```
$ gofmt hello-unformatted.go > hello-formatted.go
```

Hasilnya adalah sebagai berikut:

```go
// hello-formatted.go
package main

import "fmt"

func main() {
    fmt.Printf("halo\n") // menampilkan tulisan
    fmt.Printf("dunia")  // ini tulisan baris kedua
}
```

Bab 2. IDE untuk Go

IDE (**Integrated Development Environment**) merupakan software yang digunakan oleh pemrogram dan pengembang software untuk membangun software. IDE berisi berbagai fasilitas komprehensif yang diperlukan para pemrogram untuk membantu mereka dalam membangun software aplikasi. Secara minimal, biasanya IDE terdiri atas editor teks (untuk mengetikkan kode sumber), debugger (pencari bugs), **syntax highlighting, code completion**, serta dokumentasi / **help**. Bab ini akan membahas beberapa software yang bisa digunakan. Sebenarnya menggunakan editor teks yang menghasilkan file text / ASCII murni sudah cukup untuk bisa menuliskan dan kemudian mengkompilasi kode sumber. Pada bab ini akan dibahas Vim sebagai editor teks dan LiteIDE sebagai software IDE yang lebih lengkap untuk Go, tidak sekedar hanya untuk menuliskan kode sumber.

2.1. Menggunakan Vim

Untuk menggunakan Vim, ada plugin utama serta berbagai plugin pendukung yang bisa digunakan. Untuk instalasi berbagai plugin tersebut, ada 2 kemungkinan:

1. Jika Vim anda menggunakan Vim sebelum versi 8, gunakan Pathogen.
2. Jika Vim anda versi 8 atau lebih tinggi, gunakan **packages** dari Vim untuk **native package management**.

2.1.1. Menggunakan Pathogen

Pathogen adalah plugin dari Tim Pope yang digunakan untuk mempermudah pengelolaan plugin. Kode sumber dari Pathogen bisa diperoleh di https://github.com/tpope/vim-pathogen. Untuk instalasi, ikuti langkah berikut:

```
$ cd
$ mkdir .vim/autoload
$ mkdir .vim/bundle
$ cd .vim/autoload
$ wget -c https://raw.githubusercontent.com/tpope/vim-
pathogen/master/autoload/pathogen.vim
--2019-07-23 13:18:11--  https://raw.githubusercontent.com/tpope/vim-
pathogen/master/autoload/pathogen.vim
Resolving raw.githubusercontent.com (raw.githubusercontent.com)...
151.101.8.133
Menghubungi raw.githubusercontent.com (raw.githubusercontent.com
)|151.101.8.133|:443... terhubung.
Permintaan HTTP dikirimkan, menunggu balasan... 200 OK
Besar: 8848 (8,6K) [text/plain]
Simpan ke: `pathogen.vim'

pathogen.vim
100%[=============================================================>]
8,64K  --.-KB/s    in 0s

2019-07-23 13:18:12 (41,2 MB/s) - `pathogen.vim' disimpan [8848/8848]

$
```

Setelah itu, untuk menggunakan Pathogen, letakkan aktivasinya di `$HOME/.vimrc` atau di-copy satu direktori ke direktori `$HOME/.vim/bundle`.

2.1.2. Native Package Management

Dengan menggunakan cara ini, kita hanya perlu menyediakan direktori `$HOME/.vim/pack/default/start`, setelah itu, copy semua repo plugin ke lokasi direktori tersebut masing-masing menempati satu direktori.

2.2. Instalasi Plugin

Setelah selesai melakukan persiapan di atas, berbagai plugin yang diperlukan bisa diambil langsung dari Internet. Berikut ini adalah daftar yang digunakan penulis:

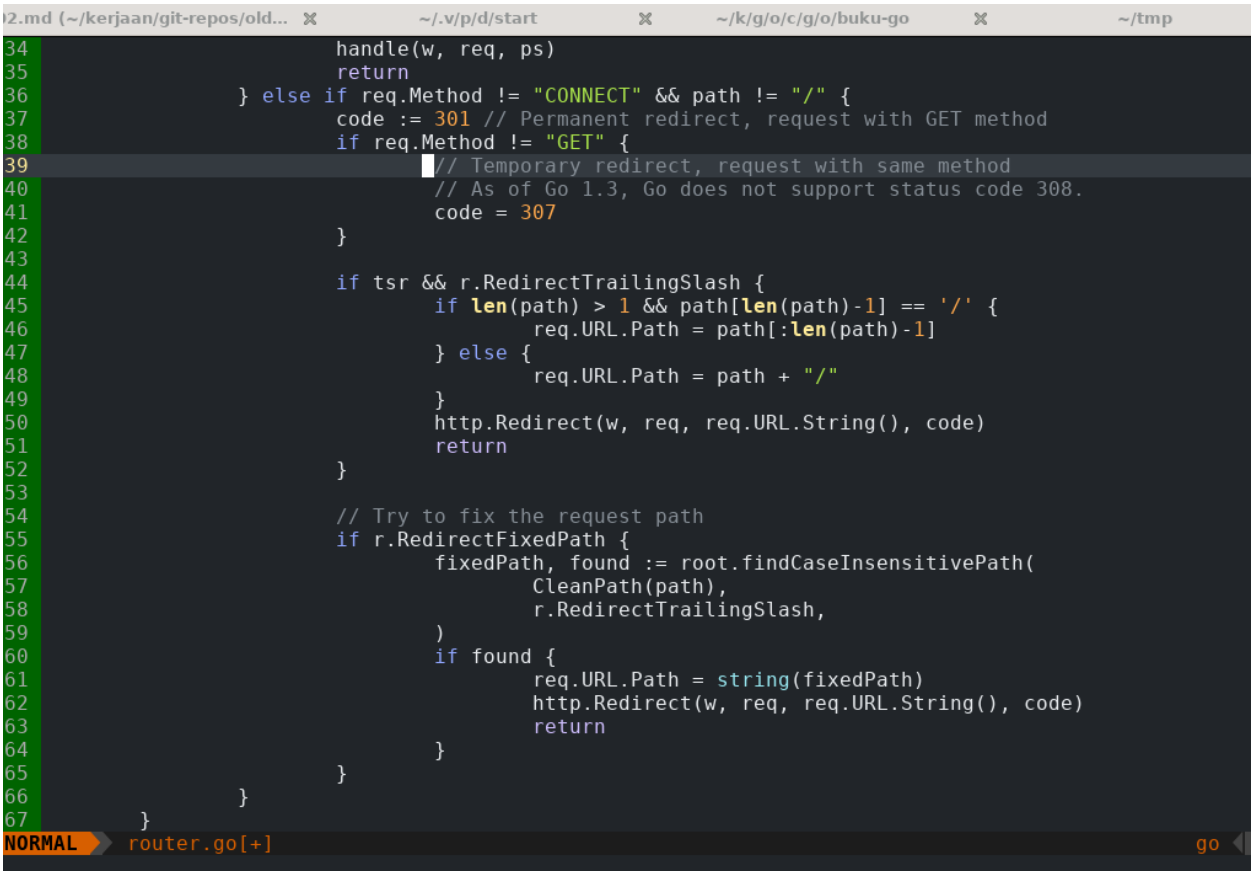
- nerdtree: untuk menampilkan file-file dalam struktur pohon di sebelah kiri sehingga memudahkan navigasi.
- nerdtree-git-plugin: untuk menampilkan status Git.
- vim-go: plugin utama agar Vim mengenali kode sumber Go.

- vim-airline: untuk menampilkan status/tabline dengan format yang lebih bagus.
- vim-airline-themes: themes dari vim-airline.
- vim-open-color: skema warna Vim menggunakan **open color**.

Cara instalasi:

```
$ cd
$ cd .vim/bundle
  atau
$ cd .vim/pack/default/start
$ git clone <masing-masing lokasi plugin>
```

Hasil dari menjalankan `vim` melalui shell untuk menulis kode sumber Go bisa dilihat pada gambar berikut ini:



2.2.1. Autocompletion

Vim menyediakan fasilitas `autocompletion` melalui `Omniautocompletion`. Fasilitas ini sudah terinstall saat kita menginstall Vim. Untuk mengaktifkan fasilitas ini untuk keperluan Go, kita

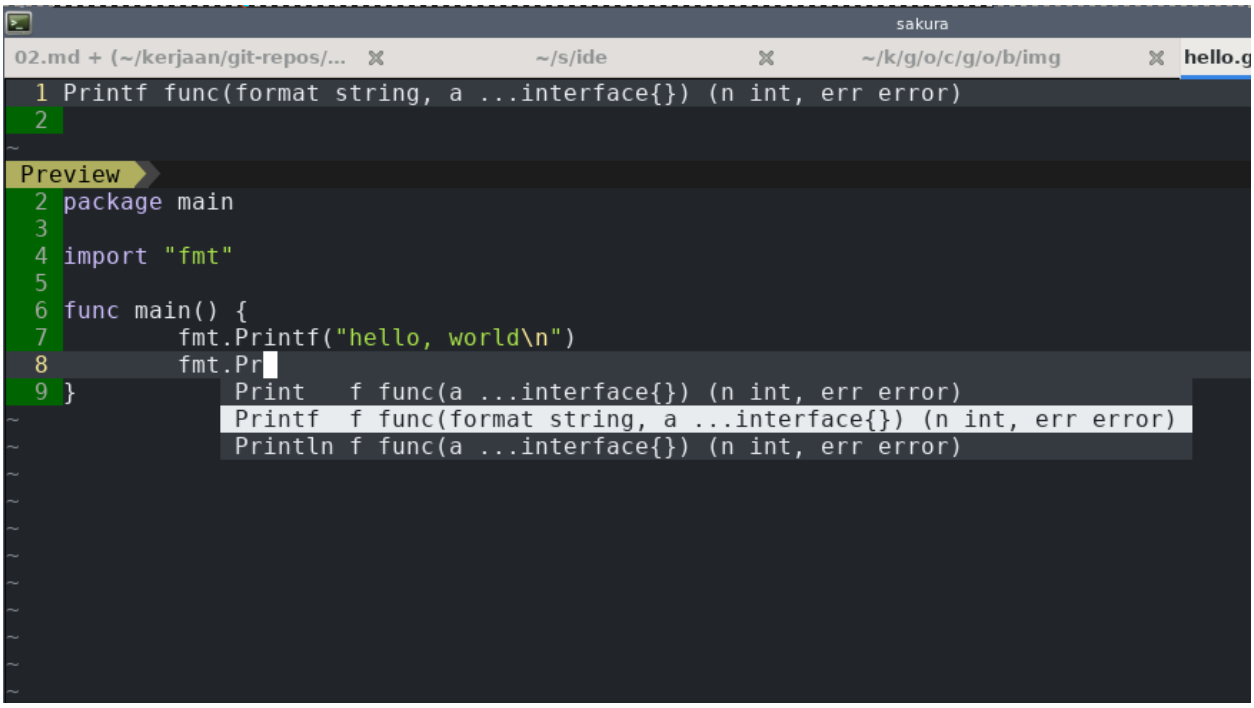
harus menginstall dan mengaktifkan gopls. Gopls sudah terinstall setelah selesai mengerjakan instalasi di bab 1. Hasil dari instalasi Gopls adalah file **binary executable** `$GOPATH/bin/gopls` (sesuai letak GOPATH di env.sh). Untuk konfigurasi, tambahkan satu baris di `$HOME/.vim/vimrc`: `set ofu=syntaxcomplete#Complete` di bawah baris `filetype plugin indent on`.

Kode sumber lengkap dari `$HOME/.vim/vimrc` yang penulis gunakan bisa dilihat pada listing berikut ini:

```
set number
set linebreak
set showbreak=+++
set textwidth=100
set showmatch
set nocompatible

set hlsearch
set smartcase
set ignorecase
set incsearch

set autoindent
set expandtab
set shiftwidth=2
set smartindent
set smarttab
set softtabstop=2

set ruler

set undolevels=1000
set backspace=indent,eol,start

filetype plugin indent on
set ofu=syntaxcomplete#Complete

" Use 24-bit (true-color) mode in Vim/Neovim when outside tmux or screen.
" If you're using tmux version 2.2 or later, you can remove the outermost
$TMUX
" check and use tmux's 24-bit color support
" (http://sunaku.github.io/tmux-24bit-color.html#usage for more
information.)
if empty($TMUX) && empty($STY)
  " See https://gist.github.com/XVilka/8346728.
  if $COLORTERM =~# 'truecolor' || $COLORTERM =~# '24bit'
    if has('termguicolors')
      " See :help xterm-true-color
      if $TERM =~# '^screen'
```

```
        let &t_8f = "\<Esc>[38;2;%lu;%lu;%lum"
        let &t_8b = "\<Esc>[48;2;%lu;%lu;%lum"
      endif
      set termguicolors
    endif
  endif
endif

set background=dark
colorscheme open-color
syntax on
highlight LineNr term=bold cterm=NONE ctermfg=DarkGrey ctermbg=NONE gui
=NONE guifg=DarkGrey guibg=darkgreen
set cursorline
" set cursorcolumn

set guifont=Monospace\ 14

" nerdtree
let g:NERDTreeWinPos = "right"
autocmd bufenter * if (winnr("$") == 1 && exists("b:NERDTree") &&
b:NERDTree.isTabTree()) | q | endif
let gRNERDTreeNodeDelimiter = "\u00a0"
nnoremap <F4> :NERDTreeToggle<CR>
let g:NERDTreeFileExtensionHighlightFullName = 1
let g:NERDTreeExactMatchHighlightFullName = 1
let g:NERDTreePatternMatchHighlightFullName = 1
let g:NERDTreeHighlightFolders = 1 " enables folder icon highlighting
using exact match
let g:NERDTreeHighlightFoldersFullName = 1 " highlights the folder name
let NERDTreeShowHidden=1

" airline
let g:airline_powerline_fonts = 1
let g:airline_theme='distinguished'
```

Untuk mengaktifkan completion, kita harus masuk ke mode `Insert` dari Vim, setelah itu tekan `Ctrl-X, Ctrl-O` secara cepat. Hasil `autocompletion` bisa dilihat di gambar berikut ini:



2.3. Menggunakan Neovim dan SpaceVim

Untuk keperluan ini, install Neovim kemudian pastikan bahwa `gopls` juga sudah terinstall (lihat bab 1). Setelah itu, gunakan SpaceVim sebagai berikut:

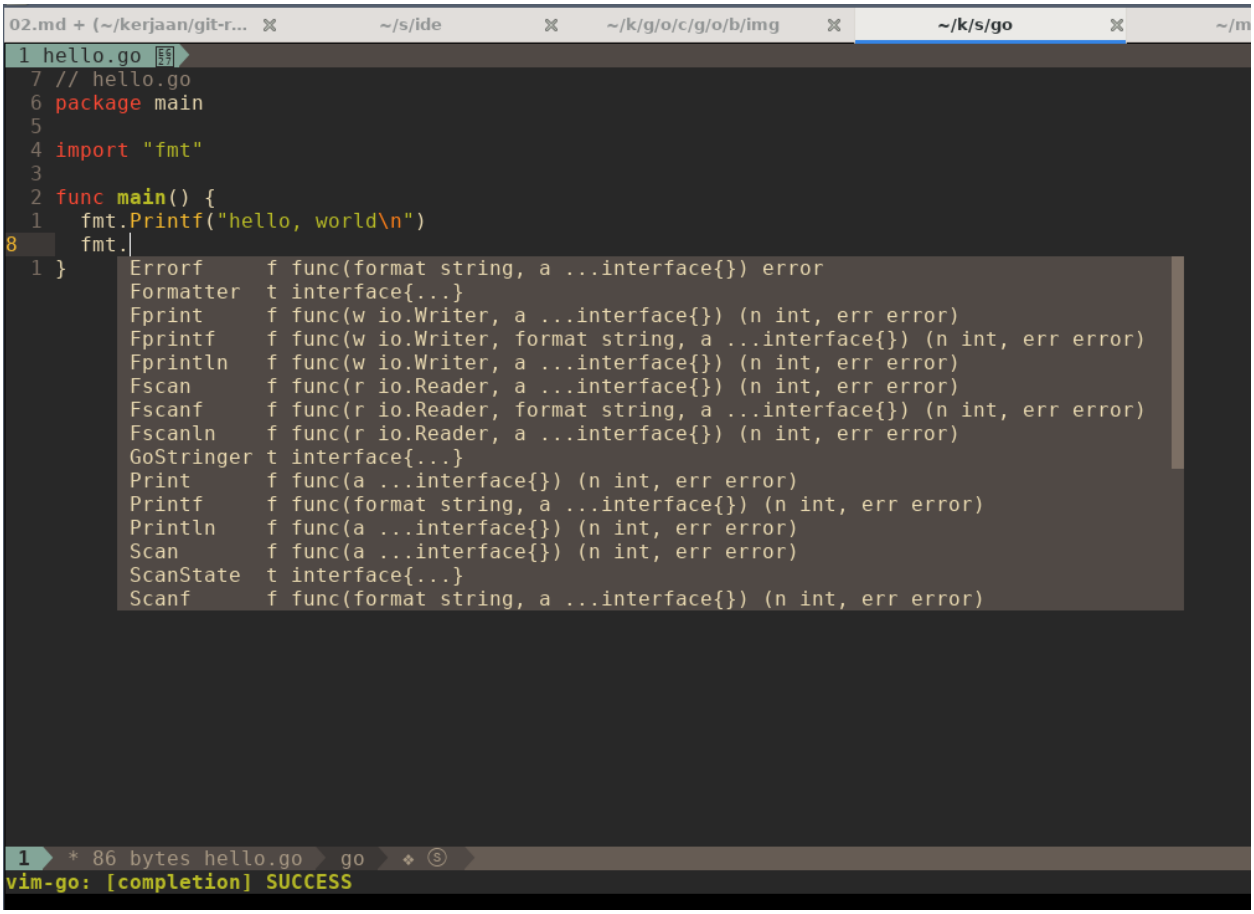
1. Clone **SpaceVim**:

```
$ cd
$ curl -sLf https://spacevim.org/install.sh | bash -s -- --install neovim
```

Hasil dari langkah di atas adalah direktori `$HOME/.SpaceVim`. Untuk update SpaceVim, lakukan `git pull` pada direktori tersebut. Untuk konfigurasi Neovim + SpaceVim sebagai IDE untuk Go, gunakan konfigurasi di `$HOME/.SpaceVim.d/init.toml` sebagai berikut:

```
#=============================================================================
====
# dark_powered.toml --- dark powered configuration example for SpaceVim
# Copyright (c) 2016-2017 Wang Shidong & Contributors
# Author: Wang Shidong < wsdjeg at 163.com >
# URL: https://spacevim.org
# License: GPLv3
#=============================================================================
====

# All SpaceVim option below [option] section
[options]
    # set spacevim theme. by default colorscheme layer is not loaded,
    # if you want to use more colorscheme, please load the colorscheme
    # layer
    colorscheme = "gruvbox"
    background = "dark"
    # Disable guicolors in basic mode, many terminal do not support 24bit
    # true colors
    enable_guicolors = true
    # Disable statusline separator, if you want to use other value, please
    # install nerd fonts
    statusline_separator = "arrow"
    statusline_inactive_separator = "arrow"
    buffer_index_type = 4
    enable_tabline_filetype_icon = true
    enable_statusline_display_mode = false

# Enable autocomplete layer
[[layers]]
name = 'autocomplete'
auto-completion-return-key-behavior = "complete"
auto-completion-tab-key-behavior = "smart"

[[layers]]
name = 'shell'
default_position = 'top'
default_height = 30

[[layers]]
  name = "lang#go"

[[layers]]
  name = "format"
```

Hasil dari konfigurasi di atas untuk proses edit kode sumber Go adalah sebagai berikut:



2.4. Menggunakan LiteIDE

LiteIDE dibuat oleh visualfc dan tersedia dalam bentuk kode sumber maupun binary. Kode sumber bisa diperoleh di https://github.com/visualfc/liteide. Installer executable bisa diperoleh di http://sourceforge.net/projects/liteide/files.

Instalasi di Linux sangat mudah, hanya tinggal mengekstrak file yang kita download pada suatu in menjalankan cukup dengan mengeksekusi file `$LITEIDE_HOME/bin/liteide` (`cd $LITEIDE_HOME/bin; ./liteide &`)

2.5. Software IDE Lain

Vim dan LiteIDE hanyalah beberapa peranti yang bisa digunakan oleh pengembang. Distribusi

Go juga menyediakan dukungan untuk berbagai peranti lunak lain:

- Emacs. Dukungan untuk Go diwujudkan dalam fasilitas `add-on`. Untuk Emacs 24 ke atas, bisa diinstall melalui manajer paket (M-x package-list-packages), cari dan install `go-mode`. Emacs juga mendukung `gopls` untuk `completion`.
- Eclipse. Dukungan untuk Go diwujudkan melalui plugin `goclipse`, bisa diperoleh di https://code.google.com/p/goclipse/.
- Selain software-software yang telah disebutkan, rata-rata IDE / Editor sudah mempunyai dukungan terhadap bahasa pemrograman Go (JEdit, Sublime-text, Notepad++, dan lain-lain).
- Visual Studio Code mempunyai dukungan yang kuat untuk Go dengan menggunakan ekstensi **Go for Visual Studio Code** - https://marketplace.visualstudio.com/items?itemName=ms-vscode.Go.

Bab 3. Struktur Program Go

Secara umum, teknik penulisan kode sumber pada Go ini harus dipahami terlebih dahulu sebelum memulai **coding**. Go telah melalui berbagai teknik penulisan kode sumber. Sampai saat ini, kita bisa memisahkan menjadi 2 bagian besar:

1. Tanpa **modules** (Go sebelum 1.11)
2. Menggunakan **modules** (Go 1.11 ke atas dengan transisi pada Go 1.11 dan Go 1.12, mulai menggunakan **modules** secara penuh pada versi 1.13).

Pada awalnya, Go menggunakan variabel lingkungan $GOPATH untuk mengelola proyek. Biasanya - seperti terlihat pada bab awal buku ini - $GOPATH berisi direktori workspace tempat **developer** menuliskan kode sumber. Jika kode sumber sudah cukup kompleks dan melibatkan banyak pustaka internal maupun eksternal, maka GOPATH ini perlu diatur, jika sederhana (hanya 1 **binary executable** tanpa pustaka internal maupun eksternal), maka $GOPATH tidak perlu diatur. Ketentuan untuk $HOPATH ini adalah sebagai berikut:

1. Lokasi pembuatan kode sumber disebut **workspace**.
2. Setiap **workspace** berisi direktori `bin`,`pkg`, dan `src`.
3. Jika $GOPATH tidak ditetepkan, maka workspace default akan berada di $HOME/go
4. Jika $GOPATH ditetapkan, **workspce** akan berada pada nilai dari $GOPATH.

Isi dari **workspace** adalah sebagai berikut:

```
$ ls -la
total 24
drwxr-xr-x 5 bpdp bpdp 4096 Jan  7  2019 ./
drwxr-xr-x 5 bpdp bpdp 4096 Jul 19 04:48 ../
drwxr-xr-x 2 bpdp bpdp 4096 Jan  7  2019 bin/
-rwxr-xr-x 1 bpdp bpdp   50 Jan  7  2019 env.sh
drwxr-xr-x 2 bpdp bpdp 4096 Jan  7  2019 pkg/
drwxr-xr-x 2 bpdp bpdp 4096 Jan  7  2019 src/
$
```

Isi dari `env.sh` adalah:

```
export GOPATH=`pwd`
export PATH=$PATH:$GOPATH/bin
```

Penjelasan masing-masing direktori tersebut:

- bin: berisi hasil kompilasi aplikasi
- pkg: berisi hasil kompilasi pustaka
- src: kode sumber untuk pustaka serta aplikasi

3.1. Persiapan

Untuk mempelajari bab ini, ada beberapa kode sumber yang harus disiapkan. Kode sumber ini akan ditulis disini lebih dahulu supaya memudahkan untuk diacu pada pembahasan berikutnya.

Nama file: `showuserenv.go`

```go
/*
    showuserenv.go

    Contoh program sederhana untuk menjelaskan
    struktur program Go untuk aplikasi executable

*/

// Program Go diawali dengan nama paket.
// Paket untuk aplikasi executable selalu berada
// pada paket main.
package main

// pustaka standar yang diperlukan
// Jika hanya satu:
// import "fmt"
// Jika lebih dari satu:
import (
    "fmt"
    "os"
)

// "Fungsi" merupakan satuan terintegrasi dari
// program Go, selalu diberi nama "main" untuk
// aplikasi executable.
func main() {

    // ini adalah kode sumber / program Go
    // akan dijelaskan lebih lanjut, abaikan
    // jika belum paham
    var (
        user    string
        homeDir string
    )

    user = os.Getenv("USER")
    homeDir = os.Getenv("HOME")

    fmt.Printf("Halo %s", user)
    fmt.Printf("\nHome anda di %s", homeDir)
    fmt.Printf("\n")

}
```

Nama file: showgoenv.go

```
/*
    showgoenv.go

    Contoh program sederhana untuk menjelaskan
    struktur program Go untuk lebih dari satu
        binary executable

*/

package main

import (
    "fmt"
    "os"
)

func main() {

    var (
        user    string
        goHome  string
        goPath  string
    )

    user = os.Getenv("USER")
    goHome = os.Getenv("GOROOT")
    goPath = os.Getenv("GOPATH")

    fmt.Printf("Halo %s", user)
    fmt.Printf("\nAnda menggunakan Go di %s", goHome)
    fmt.Printf("\nGOPATH anda di %s", goPath)
    fmt.Printf("\n")

}
```

Nama file: `reverse.go`

```
/*

        reverse.go
        Contoh pustaka sederhana untuk membalik kata.
    diambil dari https://golang.org/doc/code.html

*/
package stringutil

// Reverse returns its argument string reversed rune-wise left to right.
func Reverse(s string) string {
    r := []rune(s)
    for i, j := 0, len(r)-1; i < len(r)/2; i, j = i+1, j-1 {
        r[i], r[j] = r[j], r[i]
    }
    return string(r)
}
```

Nama file: hellostringutil.go

```
/*

        hellostringutil.go
        Contoh sederhana untuk menggambarkan cara menggunakan lib
        Diambil dari https://golang.org/doc/code.html

*/
package main

import (
    "fmt"
    "github.com/bpdp/stringutil"
)

func main() {
    fmt.Printf(stringutil.Reverse("Hello, World!"))
}
```

Nama file: hellorsc.go

```
package main

import (
    "fmt"
    "rsc.io/quote"
)

func main() {
    fmt.Println(quote.Hello())
}
```

Nama file: `gomtk.go`

```
package gomtk

func Add(x int, y int) int {
    return x + y
}
```

Nama file: `gomtktest.go`

```
package main

import (
    "fmt"
    adder "github.com/oldstager/gomtk"
)

func main() {
    fmt.Println(adder.Add(2, 4))
}
```

3.2. Tanpa Modules

3.2.1. Program Aplikasi Sederhana - 1 File `binary executable` Utama

Suatu aplikasi `executable` (artinya bisa dijalankan secara langsung oleh sistem operasi) mempunyai struktur seperti yang terlihat pada listing `showuserenv.go`. Untuk kasus sederhana dan tanpa ketergantungan kepada pustaka eksternal seperti ini, file bisa diletakkan dimana saja. Untuk menjalankan kode sumber tersebut, ikuti langkah-langkah berikut:

Tanpa Proses Kompilasi

```
$ go run showuserenv.go
Halo bpdp
Home anda di /home/bpdp
```

Mengkompilasi Menjadi Binary Executable

```
$ go build showuserenv.go
$ ls -la
total 1992
drwxr-xr-x 2 bpdp bpdp    4096 Jul 29 06:41 ./
drwxr-xr-x 3 bpdp bpdp    4096 Jul 29 06:36 ../
-rwxr-xr-x 1 bpdp bpdp 2027001 Jul 29 06:41 showuserenv*
-rw-r--r-- 1 bpdp bpdp     813 Jul 29 06:39 showuserenv.go
$ ./showuserenv
Halo bpdp
Home anda di /home/bpdp
$ file showuserenv
showuserenv: ELF 64-bit LSB executable, x86-64, version 1 (SYSV),
statically linked, Go BuildID
=9rxlkJ0BRey3wgq8FCzN/hTjP17z9sr3yZTvx1JEZ/KJMv7CpJpm_WaUyomyhS/rvGW2o0cNy
_kmNe0caJe, not stripped
$ strip showuserenv
$ ls -la
total 1408
drwxr-xr-x 2 bpdp bpdp    4096 Jul 29 06:42 ./
drwxr-xr-x 3 bpdp bpdp    4096 Jul 29 06:36 ../
-rwxr-xr-x 1 bpdp bpdp 1428296 Jul 29 06:42 showuserenv*
-rw-r--r-- 1 bpdp bpdp     813 Jul 29 06:39 showuserenv.go
$ ./showuserenv
Halo bpdp
Home anda di /home/bpdp
$
```

3.2.2. Program Aplikasi: Lebih dari 1 File `binary executable` tanpa ketergantungan pustaka eksternal

Jika tanpa pustaka internal maupun eksternal, maka membangun lebih dari satu **binary executable** dilakukan cukup dengan meletakkan pada sembarang direktori dan mem-**build** satu persatu.

```
$ ls -la
total 16
drwxr-xr-x 2 bpdp bpdp 4096 Jul 29 06:52 ./
drwxr-xr-x 4 bpdp bpdp 4096 Jul 29 06:51 ../
-rw-r--r-- 1 bpdp bpdp  503 Jul 29 06:52 showgoenv.go
-rw-r--r-- 1 bpdp bpdp  813 Jul 29 06:51 showuserenv.go
$ go build showgoenv.go
$ go build showuserenv.go
$ ls -la
total 3976
drwxr-xr-x 2 bpdp bpdp    4096 Jul 29 06:52 ./
drwxr-xr-x 4 bpdp bpdp    4096 Jul 29 06:51 ../
-rwxr-xr-x 1 bpdp bpdp 2027001 Jul 29 06:52 showgoenv*
-rw-r--r-- 1 bpdp bpdp     503 Jul 29 06:52 showgoenv.go
-rwxr-xr-x 1 bpdp bpdp 2027001 Jul 29 06:52 showuserenv*
-rw-r--r-- 1 bpdp bpdp     813 Jul 29 06:51 showuserenv.go
$
```

3.2.3. Program Aplikasi: Lebih dari 1 File `binary executable`

Untuk keperluan ini, buat **workspace** seperti petunjuk di awal bab, setelah itu, letakkan file **showuserenv.go** dan **showgoenv.go** masing-masing dalam **sub package** tersendiri. Perhatikan struktur direktori berikut:

```
.
├──── bin
│     ├──── showgoenv
│     └──── showuserenv
├──── env.fish
├──── pkg
└──── src
      ├──── showgoenv
      │     └──── showgoenv.go
      └──── showuserenv
            └──── showuserenv.go
```

Untuk nengkompilasi, **env.sh** sudah harus di-**source** terlebih dahulu. Setelah itu kompilasi sekaligus install:

```
$ go install ...
$ ls -la bin/
total 3968
drwxr-xr-x 2 bpdp bpdp    4096 Jul 29 20:43 ./
drwxr-xr-x 5 bpdp bpdp    4096 Jul 29 20:59 ../
-rwxr-xr-x 1 bpdp bpdp 2027001 Jul 29 21:01 showgoenv*
-rwxr-xr-x 1 bpdp bpdp 2027001 Jul 29 21:01 showuserenv*
$
```

3.2.4. Pustaka / Library / Package dan Penggunaannya

Ada kalanya, para software developer membangun pustaka yang berisi berbagai fungsionalitas yang bisa digunakan kembali suatu saat nanti. Untuk keperluan ini, Go menyediakan fasilitas untuk membangun library dalam bentuk kumpulan fungsi. Kumpulan fungsi ini nantinya akan diletakkan pada suatu repo tertentu sehingga bisa langsung di `go get <lokasi repo pustaka>`. Pada penjelasan berikut ini, kita akan membangun suatu aplikasi kecil (**hellostringutil**) yang menggunakan suatu pustaka yang sebelumnya sudah kita bangun (`stringutil/Reverse` - untuk membalik kata). Kode sumber diambil dari How to write Go code. Semua kode sumber, baik untuk pustaka ataupun aplikasi akan diletakkan pada pola direktori tertentu. Go menggunakan pola repo untuk penamaan / pengelompokan aplikasi atau pustaka meskipun belum dimasukkan ke repo di Internet. Sebaiknya membiasakan diri sejak awal menggunakan pola tersebut meskipun belum akan dimasukkan ke repositori di Internet. Untuk mengerjakan bagian ini, buat **workspace** terlebih dahulu.

Membuat Pustaka

Kode sumber untuk pustaka (**reverse.go**) ini akan diletakkan di `src/github.com/oldstager/stringutil`. Paket yang dibuat dengan penamaan ini, nantinya akan diacu dalam `import` sebagai `github.com/oldstager/stringutil`. Untuk mengkompilasi:

```
$ go build github.com/oldstager/stringutil
$
```

Jika tidak ada kesalahan, maka akan langsung kembali ke prompt shell.

Membuat Aplikasi yang Memanfaatkan Pustaka

Sama halnya dengan pustaka, aplikasi juga menggunakan pola penamaan yang sama. Letakkan `hellostringutil.go` di `src/github.com/oldstager/hellostringutil`.

Untuk mengkompilasi dan menjalankan:

```
$ go install ...
$ hellostringutil
!dlroW ,olleH
$ ls -la bin/
total 1976
drwxr-xr-x 2 bpdp bpdp    4096 Jul 29 21:46 ./
drwxr-xr-x 5 bpdp bpdp    4096 Jul 29 21:26 ../
-rwxr-xr-x 1 bpdp bpdp 2014199 Jul 29 21:46 hellostringutil*
$
```

3.3. Menggunakan Modules

Penggunaan **modules** lebih disarankan untuk proses pengembangan berikutnya. Saat menggunakan Go 1.11 dan Go 1.12, kita masih berada pada masa transisi. Meskipun demikian, saat **modules** telah diimplementasikan penuh di Go 1.13, tidak akan mengacaukan kode sumber dengan **modules** yang dibuat menggunakan Go 1.11 dan Go 1.12.

3.3.1. Program Aplikasi

Untuk keperluan ini, buat direktori (lokasi bebas - di luar $GOPATH). Pada direktori tersebut, inisialisasi `modules` terlebih dahulu menggunakan:

```
$ go mod init github.com/oldstager/go-to-hell-o
$ cat go.mod
module github.com/oldstager/go-to-hell-o

go 1.12
$
```

Setelah itu baru buat kode sumber `hellorsc.go` pada direktori tersebut. Untuk mengkompilasi:

```
$ go build
go: finding rsc.io/quote v1.5.2
go: downloading rsc.io/quote v1.5.2
go: extracting rsc.io/quote v1.5.2
go: finding rsc.io/sampler v1.3.0
go: finding golang.org/x/text v0.0.0-20170915032832-14c0d48ead0c
go: downloading rsc.io/sampler v1.3.0
go: extracting rsc.io/sampler v1.3.0
go: downloading golang.org/x/text v0.0.0-20170915032832-14c0d48ead0c
go: extracting golang.org/x/text v0.0.0-20170915032832-14c0d48ead0c
$
```

Hasil:

```
$ ls -la
total 2296
drwxr-xr-x 2 bpdp bpdp    4096 Jul 28 06:55 ./
drwxr-xr-x 4 bpdp bpdp    4096 Jul 28 06:45 ../
-rw------- 1 bpdp bpdp      79 Jul 28 06:55 go.mod
-rw------- 1 bpdp bpdp     499 Jul 28 06:55 go.sum
-rwxr-xr-x 1 bpdp bpdp 2327743 Jul 28 06:55 go-to-hell-o*
-rw-r--r-- 1 bpdp bpdp      93 Jul 28 06:54 hello.go
$
```

Module yang sudah di-**get** dan di-**build** berada di:

```
$ tree -L 3 ~/go/pkg/mod/
/home/bpdp/go/pkg/mod/
├── cache
│   ├── download
│   │   ├── golang.org
│   │   └── rsc.io
│   ├── lock
│   └── vcs
│       ├──
0c8659d2f971b567bc9bd6644073413a1534735b75ea8a6f1d4ee4121f78fa5b
│       ├──
0c8659d2f971b567bc9bd6644073413a1534735b75ea8a6f1d4ee4121f78fa5b.info
│       ├──
0c8659d2f971b567bc9bd6644073413a1534735b75ea8a6f1d4ee4121f78fa5b.lock
│       ├──
4db0c9594744360b0eaa452d2ccfbd45b05dffb9810882957d10d69e61e66382
│       ├──
4db0c9594744360b0eaa452d2ccfbd45b05dffb9810882957d10d69e61e66382.info
│       ├──
4db0c9594744360b0eaa452d2ccfbd45b05dffb9810882957d10d69e61e66382.lock
│       ├──
5b03666c2d7b526129bad48c5cea095aad8b83badc1daa202e7b0279e3a5d861
│       ├──
5b03666c2d7b526129bad48c5cea095aad8b83badc1daa202e7b0279e3a5d861.info
│       └──
5b03666c2d7b526129bad48c5cea095aad8b83badc1daa202e7b0279e3a5d861.lock
├── golang.org
│   └── x
│       └── text@v0.0.0-20170915032832-14c0d48ead0c
└── rsc.io
    ├── quote@v1.5.2
    │   ├── buggy
    │   ├── go.mod
    │   ├── LICENSE
    │   ├── quote.go
    │   ├── quote_test.go
    │   └── README.md
    └── sampler@v1.3.0
        ├── glass.go
        ├── glass_test.go
        ├── go.mod
        ├── hello.go
        ├── hello_test.go
        ├── LICENSE
        └── sampler.go

15 directories, 19 files
$
```

3.3.2. Pustaka dan Penggunaannya

Penggunaan `modules` menyebabkan proses pengembangan menjadi lebih sederhana. Untuk contoh ini, kita akan membuat 2 proyek:

1. `gomtk`: berisi pustaka matematika - saat ini hanya berisi 1 function, yaitu **Add**.
2. `gomtktest`: berisi program aplikasi yang akan memanggil pustaka `gomtk`.

Kedua direktori tersebut bisa berada di mana saja.

Pustaka

Direktori untuk pustaka ini diinisialisasi sebagai `module` dengan cara:

```
$ go mod init github.com/oldstager/gomtk
$ cat go.mod
module github.com/oldstager/gomtk

go 1.12
$
```

Untuk mengkompilasi:

```
$ go build
$
```

Penggunaan Pustaka

Direktori untuk penggunaan pustaka ini diinisialisasi sebagai `module` dengan cara:

```
$ go mod init github.com/oldstager/gomtktest
$
```

Aplikasi ini menggunakan pustaka, sehingga `go.mod` harus diedit:

```
module github.com/oldstager/gomtktest

go 1.12

replace github.com/oldstager/gomtk => ../gomtk
$
```

Untuk mengkompilasi menjadi **binary executable**:

```
$ go build
$ ls -la
total 1976
drwxr-xr-x 2 bpdp bpdp    4096 Jul 29 22:43 ./
drwxr-xr-x 4 bpdp bpdp    4096 Jul 28 23:43 ../
-rw------- 1 bpdp bpdp     166 Jul 28 23:45 go.mod
-rwxr-xr-x 1 bpdp bpdp 2005767 Jul 29 22:43 gomtktest*
-rw-r--r-- 1 bpdp bpdp     115 Jul 28 23:45 gomtktest.go
$
```

Part II: Sintaksis Go

Bagian ini membahas tentang sintaks dari bahasa pemrograman Go.

Bab 4. Sintaksis Dasar Go

4.1. Sub Section 1

Bab 5. Fungsi

5.1. Sub Section 1

Bab 6. Penanganan Kesalahan

6.1. Sub Section 1

Bab 7. Struktur Data

7.1. Sub Section 1

Bab 8. Konkurensi dan Paralelisme

8.1. Sub Section 1

Part III: Go untuk Kasus Spesifik

Bagian ini membahas tentang penggunaan Go untuk menyelesaikan masalah tertentu dalam pembuatan software. Penyelesaian masalah bisa dilakukan dengan menggunakan pustaka standar maupun pustaka pihak ketiga. Pada bagian ini, pembahasan tidak hanya ditujukan pada pustaka standar tetapi juga pustaka pihak ketiga selama bisa digunakan untuk menyelesaiakn masalah dalam domain masalah tertentu.

Bab 9. Testing

9.1. Sub Section 1

Bab 10. I/O dan FIlesystems

10.1. Sub Section 1

Bab 11. Akses Basis Data

11.1. Sub Section 1

Bab 12. Sistem Terdistribusi

12.1. Sub Section 1

Bab 13. Aplikasi Web

13.1. Sub Section 1