Strumenti per programmare in Go

I comandi Go

L'installazione di Go include vari strumenti per compilare, eseguire e formattare il codice, oltre ad altri strumenti di utilità.

- `go fmt`: formatta il codice di un singolo file o di un intero package
- `go run <programma.go>`: compila ed esegue un programma Go
- `go build`: compila un package creando un file eseguibile
- `go doc`: restituisce la documentazione relativa ad un package Go o ad una funzione
- `go help <comando>`: restituisce la documentazione relativa ad un comando Go

Go fmt

```
$ ls
main.go

$ go fmt main.go
```

Codice prima della formattazione:

```
package main
import "fmt"
func main() {
fmt.Prinln("Hello world!")
    }
```

->

Codice dopo la formattazione

```
package main

import "fmt"

func main() {
    fmt.Prinln("Hello world!")
}
```

Go doc

```
$ go doc fmt.Println
package fmt // import "fmt"

func Println(a ...interface{}) (n int, err error)
  Println formats using the default formats for its operands and writes to
  standard output. Spaces are always added between operands and a newline
  appended. It returns the number of bytes written and any write error
  encountered.
```

Go run

```
$ ls
main.go

$ go run main.go
Hello world!
```

```
package main

import "fmt"

func main() {
    fmt.Prinln("Hello world!")
}
```

Go build

```
$ ls
main.go

$ go build -o Hello

$ ls
Hello main.go

$ ./Hello
Hello world!
```

```
package main

import "fmt"

func main() {
    fmt.Prinln("Hello world!")
}
```

Go help

```
$ go help build
usage: go build [-o output] [-i] [build flags] [packages]

Build compiles the packages named by the import paths,
along with their dependencies, but it does not install the results.

If the arguments to build are a list of .go files from a single directory,
build treats them as a list of source files specifying a single package.

When compiling packages, build ignores files that end in '_test.go'.

...
```