Installing Go and the Go Tools-Chain in Binary Format

Go binaries

The already compiled code that allows users to download the program on the machine without needing to compile the source code is called a **binary**.

You can download GO binaries here, where you choose the file appropriate for your OS and processor architecture: msi or zip for *Windows*, pkg for *Mac OS* and tar.gz for *Linux* and *FreeBSD*.

Installing Go binaries

The Go binary distributions assume they will be installed in **/usr/local/go** or **c:\go**, but it is possible to install them in a different location. During installation, the **GOROOT** environment variable is set by default to this *root* of the Go installation tree. For example, on Windows, GOROOT has the default value c:\go.

For Windows

Start the msi installer (double click) and follow the install wizard. The Go tree will install in **c:\Go**. Alternatively, extract the zip file (with Winzip or another popular compression tool) to **c:\Go** or another folder.

For FreeBSD and Linux

Extract the archive into **/usr/local**, creating a Go tree in **/usr/local/go** For example:

```
tar -C /usr/local -xzf go$VERSION.$OS-$ARCH.tar.gz
```

Choose the name of the archive file suitable for the installation. For instance, if you are installing Go version 1.13.3 for 64-bit x86 on Linux, the archive you want is called go1.13.1.linux-amd64.tar.gz. Typically, these commands must be run as root or through *sudo* keyword.

For Mac OS

Open the **.pkg** file and follow the prompts to install the Go tools. The package installs the Go distribution to **/usr/local/go**.

Making Go binaries system-wide available

To do that, the *bin folder* under GOROOT should be available in the PATH system environment variable. In Windows and Mac OS, this is taken care of by the installation process (any open Terminal sessions must be restarted for the changes to take effect). In Linux, you must do this manually: add the following line to your **/etc/profile** (for a system-wide installation) or **$HOME/.profile**.

```
export PATH=$PATH:/usr/local/go/bin
```

or using GOROOT:

```
export PATH=$PATH:$GOROOT/bin
```

Installing to a custom folder

If you installed Go to another directory, say **d:\golang** or **$HOME/goinstall**, you must explicitly set GOROOT to that value to use the Go tooling. For example, with Linux and Mac OS, if you installed Go to your home directory ($HOME is /home/user1 if you are logged in as user1), you should add the following commands to $HOME/.profile:

```
export GOROOT=$HOME/goinstall # only to be set if different from default

export PATH=$PATH:$GOROOT/bin
```

After the changes, *restart terminal* or *reload .profile* with: source .profile and test the values with **env | grep GO** meaning *or* **echo $GOROOT** in a terminal window.

In Windows, change the GOROOT variable to your installation folder and add the path to the bin folder to PATH in the Environment Variables from System Configuration. To create or change an environment variable, do the following:

- Open Windows Explorer
- Select Computer
- Right Mouse Click: Properties -> Advanced System Settings ->Environment Variables ->System Variables -> New (or Edit).

Testing the installation

Using your favorite text editor, make a file with the following code, and save this as **hello_world1.go**.

```go
package main

import "fmt"

func main() {

   // Printing Hello World

   fmt.Println("Hello", "world")

}
```

Compile, link and run this Go-program with the go tool as follows:

```
go run hello_world1.go
```

which produces the output as **Hello world**

What is installed on your machine?#

You may wonder what the installation folder looks like. The structure of Go tree, as the installation folder is called under the go-root folder ($GOROOT), is:

```
README, AUTHORS, CONTRIBUTORS, LICENSE
\api                 these are data files for Go's API checker
\bin                 contains the executables go, godoc and gofmt
\doc                 tutorial programs, code walks,local documentation, logo’s, ...
\lib                 templates for the documentation
\pkg\os_arch         with os_arch e.g. linux_amd64, windows_386, windows_amd64,...
                     the object files (.a) of all the packages of the standard library
\tool\os_arch        contains the compiler, linker and other tools
\include             C/C++ header files
\src                 Go source files of all
                     packages in the standard
                     library bash-scripts and make command files
\test                all kinds of test files
```