First steps for software developers:

Golang for ARM-Linux systems

This manual is intended to serve as an introduction to the development of individual communication and automation solutions based on ARM-Linux systems in Go.

This language developed by Google is fast and easy to learn, but in contrast to script languages it is also very powerful and high-performance. It comes with a cross-compiler for ARM-Linux systems and offers a very simple workflow for programming the pure.box, the versatile communication and automation server from Wiesemann & Theis.

After installing and configuring a work environment, this tutorial shows you how to work with the Gotool using the command line with the example of a simple Hello-World program for the pure.box.

Step 1: Install Go

Go is free software and can be downloaded from the project page. Alternatively you can install Go from the package sources. After installation a few environment variables must be set and adjusted.

- **PATH**
  The environment variable PATH must have the sub-directory **bin** added in the GO directory so that Go can be run system-wide.
- **GOROOT**
  The environment variable GOROOT contains the path to the directory of the Go installation. This variable is only set if Go was installed in a different directory than the standard directory **/usr/local/go**.
- **GOPATH**
  The environment variable GOPATH contains the path to the working director of the user. This directory will later contain the sub-directories **src** for sources, **lib** for pre-compiled libraries, and **bin** for executable files.

Setting the environment variables
First set the variable GOPATH to your personal working directory. Then process the file .bashrc in your home directory and there insert for example the following line.

```
export GPOATH=~/go
```

If you have not installed Go from the packet sources, you must now set the variable GOROOT and expand the variable PATH. For practical reasons these changes are set for the entire system. Therefore, use root rights to edit the file /etc/profile and insert here both the following lines:

```
#Pfad zum Installationsverzeichnis von Go
export GOROOT=/usr/local/go
export PATH=$PATH:$GOROOT/bin
```

Install Git
To install language extensions and packages from third parties without great effort, Go uses Git, the software developed by Linus Torvalds for distributed version management. This can be downloaded from www.git-scm.com or can be installed using the package manager.

Check installation
The go tool is the Swiss army knife for Go development. To check whether the installation was successful, you can use it now to display the Go environment.

In the terminal enter the following command:

```
go env
```

An overview of all environment variables relevant to Go is now output:

```
GOARCH="amd64"
GOBIN=""
GOEXE=""
GOHOSTARCH="amd64"
GOHOSTOS="linux"
GOOS="linux"
GOPATH="/home/martin/go"
GORACE=""
GOROOT="/usr/lib/go"
GOTOOLDIR="/usr/lib/go/pkg/tool/linux_amd64"
CC="gcc"
GOGCCFLAGS="-fPIC -m64 -pthread -fmessage-length=0 -fdebug-prefix-map=/tmp/go-build863343041=/tmp/go-build
CXX="g++"
CGO_ENABLED="1"
```

If an error message is displayed, the bin directory in **GOROOT** is probably not in the search path. Check the environment variable **PATH** using the command **echo %PATH%**. Then check whether **GOPATH** matches your working directory.

Step 2: Hello, World

Go programs are developed within the GOPATH in the directory `src`. Create here the directory `hello` and in it the file `hello.go`.

```
package main

import "fmt"

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

Line 1:
If you want to create an executable file in Go, work in the package **main**.

Line 3:
The package **fmt** provides functions for in- and output.

Line 5:
The function **main()** is the entry point to the program.

Line 6:
fmt.Println() outputs a string followed by a newline character.

To compile the program, enter the following command:

```
go install github.com/wut/tutorial/hello
```

If you have not received an error message, the translation was successful. The executable file hello.exe can now be found in the bin directory of the Go-Path.

```
./hello
Hello World!
```

Step 3: Hello pure.box - Cross-compilation for ARM-Linux

Go makes it very simple to compile applications for other target platforms. To translate the Hello World example for the pure.box, you must simply set a few temporary environment variables:

- **GOARCH**
  The environment variable GOARCH sets the target architecture. For the pure.box this is **arm**.
- **GOOS**
  This is the operating system of the target platform. For the pure.box this variable must be set to **linux**.
- **GOARM**
  GOARM tells the compiler how it should handle floating point operations. The value for the pure.box is **5**.

Enter the following in the entry prompt:

```
export GOOS=linux
export GOARCH=arm
export GOARM=5
```

Start the cross-platform translation by changing to the source code directory and there running **go build**:

```
go build
ls -al
total 1476
drwxr-xr-x 2 gopher users    4096 Dec 25 11:31 .
drwxr-xr-x 3 gopher users    4096 Dec 25 12:08 ..
-rwxr-xr-x 1 gopher users 1496747 Dec 25 11:31 hello
-rw-r--r-- 1 gopher users      88 Dec 25 12:09 hello.go
```

As you can see, the working directory not contains the executable file **hello**. Cross-compilation with Go is just this easy!

Step 4: Run on the pure.box

There are various ways to transfer the binary to the box: In addition to access via FTP, the box can be directly incorporated into your Windows network via SMB.

Since the simplest way to try out an application is to manually run it in a terminal session, SSH should be enabled on the box. This makes data transmission via SCP obvious.

If you have no yet installed an SSH client, install the package open-ssh. For SCP you will need the following parameters: The **user name** is "admin", the **password** is the administratorpassword you set. As **host name** specify host name or the IP address of your pure.box.

Copy file:

```
scp hello admin@hostname:/data/userfiles
```

Run program:
Connect to the pure.box with Putty or using the ssh-command.

```
ssh admin@hostname
```

Then run the file **hello** in the directory **/data/userfiles**.

```
./hello
Hello pure.box
```

Still have questions? Just call us!

Do you still have questions about this guide? Our Support group will be glad to help you!

We are available to you in person:

Wiesemann & Theis GmbH
Porschestr. 12
42279 Wuppertal
Phone: +49 202/2680-110 (Mon.-Fri. 8 a.m. to 5 p.m.)
Fax: +49 202/2680-265
info@wut.de