Building and Running Go-Programs

This lesson discusses the steps required to build and run a Go program.

WE'LL COVER THE FOLLOWING

The `go` command

To compile and link a go program **test.go**, use:

```
go build test.go
```

If you want to compile, link and run the program, use:

```
go run test.go
```

where the executable binary is stored in the cache. If you want to compile, link and install the package `pack1`, use:

```
go install pack1
```

To remove any object and cache file, use:

```
go clean
```

Messages from the compiler

- If the build process (which is also called compile-time) doesn't give any errors, no message appears.
- If building produces (an) error(s), we get the following output: `---- Build`

`file exited with code 1`, and the line number where the error occurs in

the source-code is indicated, like this, if we insert var `a` at **line 6**: `hello_world.go:6: syntax error: unexpected newline, expecting type`

- In most IDEs, double-clicking the error line positions the editor on that code line where the error occurs.
- Go does not distinguish between warnings and errors. The philosophy is: when it is worth warning for misuse of something, it better be signaled as an error to be always on the safe side.
- If you try to execute a program which is not yet compiled, you get the following error: `---- Error run with code File not found, resource error`.
- When executing a program, you are in the run-time environment. If the program executes correctly, after the execution the following is the output: `---- Program exited with code 0`.

Compiling all go-files in a directory

Here is the Linux-version of a useful script for the purpose of quick testing for compilation errors in a lot of files:

```
#!/bin/bash
FILES=~/go_projects/src/*.go
for f in $FILES
do
echo "Processing $f file..."
# take action on each file. $f stores current filename
# cat $f
go build $f >> compout
done
```

You need to replace `~/go_projects/src/` with your folder name. This will compile all the go-files in the current directory and write the output to `compout`.

That's it about building and running a Go program via the command line. The next lesson discusses the *help* feature provided by Go.