Golang заметки - Кросс-платформенная компиляция

Created By	Daniil A (Kiky Tokamuro)
Last Edited	@Aug 3, 2020
Tags	golang

Очень часто бывает так, что приложение разрабатывается в одной ОС, а эксплуатироваться должно в другой, данную проблему решает кросс-платформенная компиляция.

Начиная с версии 1.5 компилятор языка Go начал поддерживать кросс-платформенную компиляцию из коробки. Для этого необходимо в переменных окружения `GOARCH` и `GOOS` указать целевую архитектуру и ОС.

`GOOS` может быть выставлена в такие значения как: android, darwin, dragonfly, freebsd, illumos, js, linux, netbsd, openbsd, plan9, solaris и windows.

`GOARCH` может быть выставлена в такие значения как: amd64 (64-bit x86), 386 (32-bit x86), arm (32-bit ARM), arm64 (64-bit ARM), ppc64le (PowerPC 64-bit, little-endian), ppc64 (PowerPC 64-bit, big-endian), mips64le (MIPS 64-bit, little-endian), mips64 (MIPS 64-bit, big-endian), mipsle (MIPS 32-bit, little-endian), mips (MIPS 32-bit, big-endian), s390x (IBM System z 64-bit, big-endian), и wasm (WebAssembly 32-bit).

Допустимыми комбинациями GOOS и GOARCH являются:

GOOS	GOARCH
netbsd	386
windows	arm
darwin	arm64
netbsd	amd64
android	amd64
linux	riscv64
openbsd	arm
plan9	arm

GOOS	GOARCH
solaris	amd64
linux	arm
dragonfly	amd64
aix	ppc64
linux	amd64
linux	ppc64le
linux	s390x
linux	arm64
linux	386
openbsd	386
netbsd	arm64
openbsd	arm64
freebsd	amd64
linux	mips
openbsd	amd64
illumos	amd64
darwin	arm
android	arm
darwin	amd64
freebsd	arm64
plan9	386
netbsd	arm
plan9	amd64
linux	ppc64
android	arm64
linux	mipsle
darwin	386
freebsd	386
android	386
js	wasm
windows	amd64
windows	386
freebsd	arm
linux	mips64le

Aa GOOS	GOARCH
linux	mips64

Пример использования

```
package main

import "fmt"

func main() {
  fmt.Println("Hello World")
}
```

Собираем под Windows с 386 архитектурой:

```
GOOS=windows GOARCH=386 go build main.go
```

Проверяем правильно ли собралось:

```
$ file main.exe
main.exe: PE32 executable (console) Intel 80386 (stripped to external PDB), for MS Windows
```