GO-10 03: Сборка под разные ОС

Описание:

Это задание будет посвящено одной из возможностей сборки программ на Go - Toolchain Go, которая позволяет создавать кросс-платформенные программы. Эту возможность ввели с версии языка 1.5. Причем изначально разработчики Go внесли возможность компиляции только под UNIX-системы, но так как язык - Open Source, то пользователи быстро написали возможность компиляции и для популярной операционной системы Windows. В Go 1.13 были внесены изменения, связанные с кросс-компиляцией. Например, полностью убрали поддержку Native Client (https://ru.wikipedia.org/wiki/Native_Client). Также некоторые изменения были связаны и с другими операционными системами. К примеру, минимальной для Mac OS стала версия El Capitan и позднее, более ранние версии не поддерживаются. Также не поддерживаются версии, вышедшие раньше Windows 7.
Создание программ под разные ОС:
Для того чтобы создавать код под разные операционные системы, Go предоставляет ряд простейших инструментов. Давайте разберемся, как создавать простейшие программы для разных ОС.
Прежде чем собирать наш код под разные системы, требуется определить, под какие платформы go может компилировать код. Инструменты Go имеют команду, которая может вывести список поддерживаемых ЯП платформ для сборки. Этот список может меняться с каждым новым релизом Go, так что описанные здесь комбинации могут отличаться в разных версиях. Чтобы просмотреть список поддерживаемых платформ, запустите следующую команду:

```
go tool dist list
```

Данный вывод — это набор пар ключ-значение, разделенных /. Первая часть комбинации перед символом / — это операционная система. В Go эти операционные системы — это возможные значения для переменной среды GOOS, произносится “goose”, название которой означает - операционная система Go. Вторая часть, идущая после /, — это архитектура. Как и в случае с операционной системой, это все возможные значения для переменной среды: GOARCH. Это сокращение произносится “gore-ch” и означает - архитектура Go.
Первая часть этого вывода - это операционная система, вторая - вид процессора, под который будет собран наш код. По умолчанию компилятор Go собирает весь код под вашу операционную систему.
Для сборки под определенную систему требуется указать переменные среды (env переменные). К примеру, если мы хотим скомпилировать наш код под операционную систему Mac OS, нам требуется указать:

```
export GOOS = darwin
export GOARCH = amd64
```

При каждой компиляции полученный бинарный файл будет компилироваться сразу под ОС, указанную в переменных среды. К сожалению, такой вариант подходит не для всего и вынуждает разработчика каждый раз менять переменные среды. Как же скомпилировать файл одной командой и без лишних импортов переменных в нашу операционную систему? Тут на помощь приходит следующая конструкция:

```
GOOS=linux GOARCH=amd64 go build main.go
```

Немного о CGO

На самом деле, некоторые стандартные библиотеки зависят от библиотек языка C, что приводит к ряду проблем при сборке. Код на некоторых машинах может не заработать и выдавать ошибки. Почему так происходит и как Go зависит от библиотек C, вы узнаете в дальнейших модулях. Для того чтобы собрать код без использования зависимостей от C, требуется указать специальный флаг.

```
GOOS=linux GOARCH=amd64 CGO_ENABLED=0 go build main.go
```

Вот небольшой список файликов, в которых содержится зависимость от C:

- crypto/x509/root_cgo_darwin.go
- net/cgo_android.go
- net/cgo_linux.go
- net/cgo_netbsd.go
- net/cgo_openbsd.go
- net/cgo_unix_test.go
- os/user/lookup_unix.go
- runtime/crash_cgo_test.go

Теги сборки под разные ОС:

Иногда недостаточно собрать файлик просто с глобальными переменными, так как требуется переопределить реализацию в самом коде. Для этого можно использовать build tags. Перейдем к примеру. В пакете os можно заметить много файлов с одинаковым названием, но разным префиксом. Например, можно заметить файлы следующих названий: os/path_unix.go,os/path_windows.go. При внимательном рассмотрении мы заметим, что в этих файлах разные build tags.

```
//path_windows.go
. . .
package os

const (
        PathSeparator     = '\\' // OS-specific path separator
        PathListSeparator = ';'  // OS-specific path list separator
)
. . .

//path_unix.go
```

```
. . .
// +build aix darwin dragonfly freebsd js,wasm linux nacl netbsd
openbsd solaris
package os

const (
        PathSeparator     = '\\' // OS-specific path separator
        PathListSeparator = ':'  // OS-specific path list separator
)
. . .
```

Built tags отключают сборку данного файла при сборке пакета. Таким образом, при сборке файла для unix файл path_windows.go не будет включен в сборку и сепараторы будут равны:

```
const (
        PathSeparator     = '\\' // OS-specific path separator
        PathListSeparator = ':'  // OS-specific path list separator
)
```

Примечания:

В основном Go разработчики компилируют под несколько систем. Это Mac OS, Windows и Linux. Но также Go поддерживает и другие ОС, например, android. В действительности, возможно собрать код, работающий под Android, но это потребует определенных действий от пользователя и просто значениями переменных тут уже не отделаешься.

Задание:

1. Форкните репозиторий module10_03 с кодом данного задания в группу с вашими репозиториями - golang_users_repos/<your_gitlab_id>.
2. В вашем проекте module10_03 сделайте новую ветку module10_03.
3. В данном репозитории содержится простейший код в виде Hello + text. Сейчас этот код всегда выводит текст "Hello MacOS". Ваша задача - сделать так, чтобы код выводил "Hello" + любая из 3 операционных систем: Linux, MacOS, Windows (в зависимости от собираемой версии).
4. В ответ на задание пришлите:

- ссылку на merge request в ветку master вашего проекта ветки module10_03;
- три команды, компилирующие код под три операционных системы.