build

golang build 交叉编译

补充说明

在很多时候，由于开发的方便，会有这样的场景出现，使用Mac开发或使用Windows开发，需要编译成Linux系统的执行文件，那么如何做到Go语言提供了非常方便的命令行操作，即可实现。

1.Mac下编译Linux, Windows

```
# Linux
CGO_ENABLED=0 GOOS=linux GOARCH=amd64 go build filename.go
# Windows
CGO_ENABLED=0 GOOS=windows GOARCH=amd64 go build filename.go

如: CGO_ENABLED=0 GOOS=windows GOARCH=amd64 go build -o helloworld-windows
helloworld.go
```

2.Linux下编译Mac, Windows

```
# Mac
CGO_ENABLED=0 GOOS=darwin GOARCH=amd64 go build filename.go
# Windows
CGO_ENABLED=0 GOOS=windows GOARCH=amd64 go build filename.go
```

3.Windows下编译Mac, Linux

```
# Mac
SET CGO_ENABLED=0
SET GOOS=darwin
SET GOARCH=amd64
go build filename.go

# Linux
SET CGO_ENABLED=0
SET GOOS=linux
SET GOARCH=amd64
go build filename.go
```

4.参数说明

```
查看环境：

$> go env
GO111MODULE=""
GOARCH="amd64"
GOBIN=""
GOCACHE="/Users/chanix/Library/Caches/go-build"
GOENV="/Users/chanix/Library/Application Support/go/env"
```

```
GOEXE=""
GOFLAGS=""
GOHOSTARCH="amd64"
GOHOSTOS="darwin"
GONOPROXY=""
GONOSUMDB=""
GOOS="darwin"
GOPATH="/Users/chanix/go"
GOPRIVATE=""
GOPROXY="https://proxy.golang.org,direct"
GOROOT="/usr/local/go"
GOSUMDB="sum.golang.org"
GOTMPDIR=""
GOTOOLDIR="/usr/local/go/pkg/tool/darwin_amd64"
GCCGO="gccgo"
AR="ar"
CC="clang"
CXX="clang++"
CGO_ENABLED="1"
GOMOD=""
CGO_CFLAGS="-g -O2"
CGO_CPPFLAGS=""
CGO_CXXFLAGS="-g -O2"
CGO_FFLAGS="-g -O2"
CGO_LDFLAGS="-g -O2"
PKG_CONFIG="pkg-config"
GOGCCFLAGS="-fPIC -m64 -pthread -fno-caret-diagnostics -Qunused-arguments -
fmessage-length=0 -fdebug-prefix-
map=/var/folders/ln/v9zzjzys4rsbfnsw4_8l2cgm0000gn/T/go-
build896214833=/tmp/go-build -gno-record-gcc-switches -fno-common"
```