#### Go开发环境工具链操作指南

1. **Go常用工具链说明**

go build 将开发人员编写的go文件编译成可执行的二进制文件

go mod 三方包管理

gofmt Go语言代码格式化格局

goimports 检查代码包导入工具

golint 检查代码规范工具

go vet 代码静态分析工具

go test 运行测试用例

1. **工具安装指导**

前提：已正确安装并配置go语言环境（详见：Go开发环境搭建操作指南）

go build、go mod、gofmt、go vet、go test 在go 语言环境已内置

go lint、goimports需要下载源码安装，操作如下：

1. . 创建安装目录

mkdir -p $GOPATH/src/golang.org/x/

2）. 下载安装包

git clone <https://github.com/golang/lint.git>

git clone <https://github.com/golang/tools.git>

3）. 安装go lint工具

go install $GOPATH/src/golang.org/x/golint/

cp $GOPATH/bin/golint /user/local/bin/

1. . 安装goimports工具

go install $GOPATH/src/golang.org/x/tools/cmd/goimports/

cp $GOPATH/bin/goimports /user/local/bin

1. **工具使用说明**

**go build 将go文件编译成可执行的二进制文件**

执行：go build xxx.go/dir // 在当前目录下输出编译生成的二进制文件，

go build -o test . // 名称默认是目录名，可通过-o 参数指定二进制名称

-gcflags ”-N -l -m” -N:禁止优化 -l:禁止内联 -m:输出优化信息

-ldflags “-w -s” -w:禁用DRWA调试信息 -s:禁用符号表

GO111MODULE=on GOPROXY="https://mirrors.aliyun.com/goproxy" go build -o test . // 直接在编译命令中指定编译模式，镜像管理地址（也可在全局go env中指定）

GOOS=linux GOARCH=amd64 go build -o test . // 交叉编译

**go mod** **三方包管理**

执行：go mod init // 生成go.mod 文件

go mod tidy // 通过GOPROXY 源拉取三方包，放在$GOPATH/pkg/mod下，并生成包记录文件go.sum

go mod vendor // 会创建vendor目录导入第三方包（作用：例如需要修改第三方包，但是不影响其他项目，可以使用go vendor模式）

**gofmt 代码格式化**

执行：gofmt xxx.go/dir // 输出代码格式化之后的内容

gofmt -s xxx.go/dir // 使用-s参数可以开启简化代码功能

例如：

s := s[1:len(s)]

格式化后：

s := s[1:]

**goimports**

执行：goimports xxx.go // 输出包导入格式化之后的文件（会对包分类换行，删除无用的包等）

**golint**

执行：golint xxx.go/dir // 会对命名规范进行检查，例如定义一个包名main\_会有如下报错

root@wangbin:/home/goproject/src/athens/cmd/proxy# ls

actions Dockerfile main.go

root@wangbin:/home/goproject/src/athens/cmd/proxy# golint

main.go:1:1: don't use an underscore in package name

main.go:13:2: a blank import should be only in a main or test package, or have a comment justifying it

**go vet**

执行：go vet xxx.go/dir // 会对go编码进行静态分析

例如：使用%d 打印一个string类型的变量，会有如下报错

a := "ni hao"

fmt.Printf("print %d", a)

PS E:\gospace\src\mystarter\src> go vet

# mystarter/src

.\main.go:20:2: fmt.Printf format %d has arg a of wrong type string

**go test**

// 对当前路径下的go函数执行测试用例

例如：

package controller

import (

    "testing"

)

func TestCheckHostName(t \*testing.T) {

    want := "host0"

    actual := GetHostName()

    if want != actual {

        t.Errorf("test CheckHostName failed, want %s, actual: %s", want, actual)

    }

}

PS E:\gospace\src\mystarter\src\controller> go test

--- FAIL: TestCheckHostName (0.00s)

hostname\_test.go:11: test CheckHostName failed, want host0, actual: test

FAIL

exit status 1

FAIL mystarter/src/controller 0.276s

**go test -cover**  // -cover参数可以统计覆盖率

例如：

PS E:\gospace\src\mystarter\src\controller> go test -cover

--- FAIL: TestCheckHostName (0.00s)

coverage: 40.0% of statements