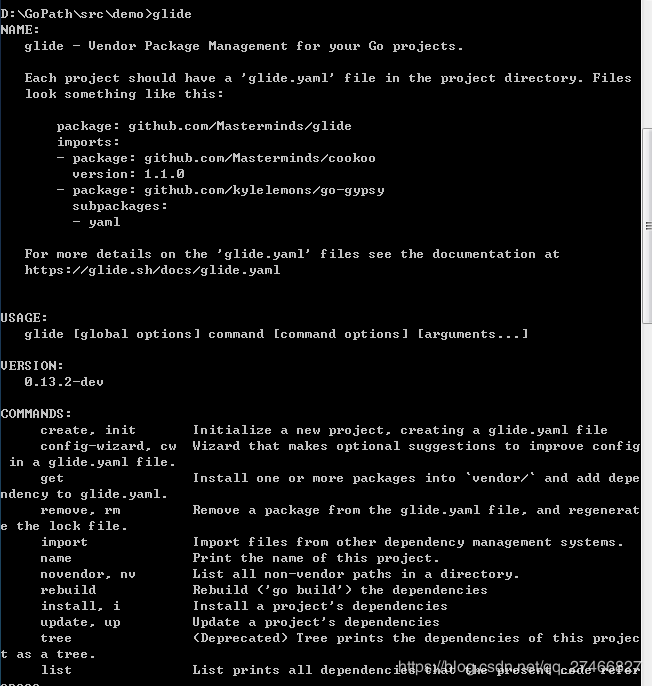
# Go使用glide包管理工具异常：Unable to export dependencies to vendor directory: Error moving files: exit status

## 1.安装glide

go get github.com/Masterminds/glide

go install github.com/Masterminds/glide

验证：输入glide,出现如下内容表示安装成功



## 2.使用glide命令

①、初始化配置  
glide create (别名 init)  
②、下载包  
glide get [package name]

例如：

glide get github.com/Masterminds/cookoo

或传递版本号

lide get github.com/Masterminds/cookoo#^1.2.3

glide update (别名 up)：

它还将递归遍历依赖包以获取任何所需的配置并在任何配置中读取。

glide install：

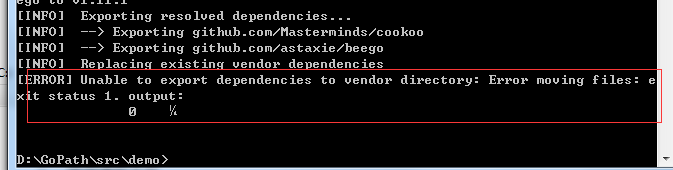
当你想从glide.lock文件安装特定的版本使用glide install

这将读取glide.lock文件，警告你如果它没有绑定到glide.yaml文件，并安装commit id特定的版本。

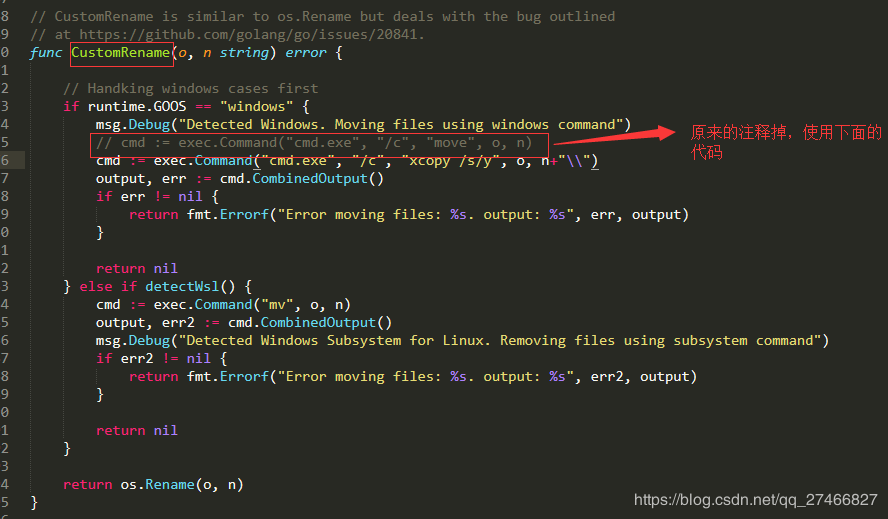
当glide.lock文件不绑定到glide.yaml文件时，如有更改，它将提供警告。运行glide up将在更新依赖关系树时重新创建glide.lock文件。

如果没有glide.lock文件存在glide install将执行update并生成 lock 文件。

## 3.使用glide get 命令安装一个包的时候会出现如下错误：



此时找到GOPATH目录下  
%GOPATH%\src\[github.com](http://github.com/)\Masterminds\glide\path文件下的winbug.go文件  
修改该文件：



func CustomRename(o, n string) error {

// Handking windows cases first

if runtime.GOOS == "windows" {

msg.Debug("Detected Windows. Moving files using windows command")

// cmd := exec.Command("cmd.exe", "/c", "move", o, n)

cmd := exec.Command("cmd.exe", "/c", "xcopy /s/y", o, n+"\\")

output, err := cmd.CombinedOutput()

if err != nil {

return fmt.Errorf("Error moving files: %s. output: %s", err, output)

}

return nil

} else if detectWsl() {

cmd := exec.Command("mv", o, n)

output, err2 := cmd.CombinedOutput()

msg.Debug("Detected Windows Subsystem for Linux. Removing files using subsystem command")

if err2 != nil {

return fmt.Errorf("Error moving files: %s. output: %s", err2, output)

}

return nil

}

return os.Rename(o, n)

}

重新编译glide即重新执行go install [github.com/Masterminds/glide命令](http://github.com/Masterminds/glide%E5%91%BD%E4%BB%A4)  
重新下载包即可解决!