对 golang 拔插程序的初级体验

作者：xhaoxiong
原文链接：https://ld246.com/article/1557134861512
来源网站：链滴

最近在帮朋友做毕设的一部分,主要需求是:可拔插的程序进行服务器的重启部署等等

最初思路是:一个服务器用作配置进行编译.so文件

其他子服务器调用此服务器上的配置好的.so服务

编译部分:

```
_, err := exec.Command("go", "build", "-buildmode=plugin", "-o", "plugin/"+serviceName+".s
",
        "plugin/"+serviceName+".go").Output()
```

编译完成后:写入消息进入消息队列

调用阶段:

```
type Common interface {
    Service([]byte) []byte
}
    open, err := plugin.Open($addr+ $serviceName + ".so")

    if err != nil {
        fmt.Println("打开插件错误:", err)
        return nil
    }

    symbol, err := open.Lookup(serviceName)
    if err != nil {
        fmt.Println("服务未发现:", err)
        return nil
    }

    if srv, ok := symbol.(Common); ok {
        return srv.Service(data)
    }

    return nil
```

是的,调用的时候报错了 not implement 不能远程通过网络链接调用，换了一种方式下载到了本地，后再调用，可以使用，但是本地服务器和远端服务器系统环境必须一样，否则不能执行。 后来摒弃了态链接拔插的这种方式了。接触不多且打开方式不正确，就放弃了

源码中的例子

```
// A Symbol is a pointer to a variable or function.
//
// For example, a plugin defined as
//
//    package main
//
//    import "fmt"
//
```

原文链接：对 golang 拔插程序的初级体验

```
//    var V int
//
//    func F() { fmt.Printf("Hello, number %d\n", V) }
//
// may be loaded with the Open function and then the exported package
// symbols V and F can be accessed
//
//    p, err := plugin.Open("plugin_name.so")
//    if err != nil {
//        panic(err)
//    }
//    v, err := p.Lookup("V")
//    if err != nil {
//        panic(err)
//    }
//    f, err := p.Lookup("F")
//    if err != nil {
//        panic(err)
//    }
//    *v.(*int) = 7
//    f.(func())() // prints "Hello, number 7"
```