链滴

尝试修复了 Go1.10 之后 gocode 完全不能使用的问题

作者：localvar

原文链接：https://ld246.com/article/1528000131328

来源网站：链滴

Go1.10中的 go build 引入了缓存机制，导致 gocode 完全不能使用（gocode#500），bug报了好个，但作者表示目前太忙没有时间修复。

Matthew Dempsky 的 fork 使用源码解析的方式解决了问题，但性能太差，不论项目大小，每次都好几秒才能给出自动完成提示。他的另一个分支，尝试从编译缓存中提取信息来提高性能，但前提是必须被编译过，我试了一下，感觉不好用。

但没自动提示写代码太费劲了，遂决定自己动手，丰衣足食。先是找到了这篇文档，读了一遍觉得太杂，短时间内啃不动，就再找投机取巧的方法，发现源码解析的主要逻辑都在 go/internal/srcimporter 这，而这个包代码量不大，细读之后更发现它已经使用了缓存，只不过这个缓存绑到了 importer 例上，不同实例不能共享。所以，基于 srcimporter 的代码，把缓存提升到全局级别即可。

代码放到了 https://github.com/localvar/gocode，需要的请自取。自己随便测了测，没发现大问，第一次自动提示时由于缓存是空的，会比较慢，后面缓存建立起来了就好多了。

本来想着建个 PR 把改动提交给 Matthew Dempsky，但觉得我的改法只是个临时方案（srcimporter 是内部包，但我把它的代码复制了一份出来修改，如果 Go1.11 有什么变动，这个方案就又废了），放弃。

已知问题：

1. 不支持 Cgo，原始的 srcimporter 就不支持，所以这个不是我的错。

2. 每个自动提示请求中都有个 context，其中包含了当前使用的 GOOS, GOARCH等信息，context 同，源码的解析结果就可能有差别。使用全局包缓存后，会出现缓存的包是基于 context A 建立的而求要求使用 context B的情况，这可能导致一些错误。不过，我觉得 context 切换的频率应该不高，以问题不严重。