红尘里的Haskell（之一）——Haskell工具链科普

Felis sap...
函数式编程、编程语言、编程 话题的优秀回答者

已关注

受@大魔头-诺铁邀请，我在这个专栏会写一些偏应用向的Haskell相关教程，内容大致参考24 Days of Hackage和What I Wish I Knew When Learning Haskell，标题就取作老司机带逛Hackage，不对，红尘里的Haskell好了（仙境里的部分还是大魔头自己写）~

第一期先科普一下Haskell工具链的一些基础知识。这些在Haskell相关的课程和教材里一般着墨甚少，而且不一定符合最新的开发实践。为了顺利配置开发环境和爽快撸码，这点微小的工作也是很重要的。

首先说说Haskell语言与实现。Haskell有语言标准：Haskell Language Report，目前出了98和2010两版，后者变化不大，有少量增补。这个语言标准里规定的是一个“经典”的Haskell语言，基于ML家族语言常用的Hindley-Milner类型系统，加入了type class特性，默认惰性求值，标准库里规定IO monad管理副作用。这个“经典”Haskell也正是大多数入门Haskell课程与教材所覆盖的范围。

Haskell有多个实现，具体可查阅Haskell wiki。我们重点关注的是GHC，因为这个项目已经是Haskell语言的事实标准。它实现了大量Language extension，在98/2010标准的基础上极大丰富了语言特性，同时也是学界前沿成果的集散地。一般而言，社区谈论的Haskell，特指GHC Haskell而非标准Haskell。

GHC是一个Haskell编译器，有编译模式和解释模式。在编译模式下，将输入Haskell代码进行一系列变换，通过assembler后端或LLVM后端生成可链接的原生机器码。在解释模式下，生成ghci

讲到这里，最简单的配环境方案呼之欲出：直接用平台相关的包管理器装一发ghc，然后直接用ghc或ghci运行程序。虽然简单，不过这时程序只能使用ghc boot libraries（ghc自带库），用不上之后带逛的各路Hackage库。所以安装ghc的事暂且按下不表，先继续科普~

接下来说Module和Package。Haskell的一个源文件对应一个Module，文件名与Module层级对应，比如module Network.HTTP.Client.Conduit，其实现就在Network/HTTP/Client/Conduit.hs。一个Module的顶层namespace里，默认已经import了Prelude（标准库）的定义，当然可以import其他Module，最后export一些定义（默认export所有本地定义，当然也可以隐藏一些本地定义，或者re-export其他Module的定义）。

不过我们谈论Haskell“库”时，指的不是Module，而是Package。ghc的package system名叫Cabal（Common Architecture for Building Applications and Libraries），文档在Cabal User Guide。一个Cabal package有若干build target，每个target需要指明种类（library/executable/test-suite/benchmark之一）、依赖的其他Cabal package名字与版本号范围、暴露给本package的第三方使用者的Module、不暴露的Module、编译选项、可执行文件名与入口（library无需）等等。

开发者除了按Module层级组织好源文件以外，额外写一个package-name.cabal文件指明这些元数据，附上LICENSE和README等物，打包，即完成了一个Cabal package，可以上传到Hackage（Haskell社区的package集散地）供他人使用。Hackage自身除了托管这些package的tarball以外，自身还是一个CI（Continuous Integration）系统，会自动尝试编译上传的package，之后从源代码生成haddock文档页面（Haskell原生支持的文档格式）。由于非Haskell依赖等各种原因，Hackage的自动build并不可靠，所以看见某个package报告build failed时不必因此质疑package质量。值得一提的是，Hackage现在在国内已经有TUNA托管的镜像了，亲测稳定好用！使用方法参考这里。

ghc安装时自带了少量的boot libraries，这些自带库的版本号与ghc版本号一般是绑定好的，不会rebuild。在编译/解释源代码时，ghc会读取Package Database，这个database是所有注册过的Cabal package的索引。需要使用第三方库时，我们使用cabal-install或stack之类的包管理器，它们会识别所有依赖，计算出build plan，并将依赖依次编译后注册到某个合适的Package Database（cabal有全局database和sandbox database，stack则有snapshot database，之后再叙），之后即可顺利import第三方Module。Package Database不是唯一的，同一个ghc可以基于不同的database工作，这是一个很重要的设计，之后我们会看到原因。

我们已经具备了开发Haskell项目需要的基础知识，现在该落实到工具的使用了。理论上，不依赖任何包管理器，仅使用ghc-pkg等ghc自带工具来管理Haskell项目的依赖也是可以的——如果闲得蛋疼。另一种“省事”的办法是使用平台的包管理器安装Haskell包，优点是快（多数平台的包管理器分发二进制，无需编译），缺点是难以控制包版本号，也难保某个发行版的打包者一抽风搞出版本号根本不兼容的情况，而需要多个版本的编译器或库时就更难胜任。因此，强烈不推荐使用平台的包管理器安装各种Haskell包，建议使用cabal-install或者stack这样的Haskell“原生”包管理器。

要用到Hackage上新发布的包时，需要运行cabal update，更新本地的Hackage目录。

提起cabal-install，顺便黑一把cabal hell好了——cabal-install过去常导致的问题。cabal-install默认会在~/.cabal维护全局的Package Database，本地开发多个Haskell项目时，其依赖是共享的。共享依赖的问题在于，不同项目计算出的build plan可能对同一个包拥有冲突的版本号；而即使支持安装同一个包的不同版本，在链接阶段（7.8及之前的）ghc仍只识别一个版本，在菱形依赖中，一个被用到不同版本的base package将导致无法编译。cabal hell难以提防的另一地方在于，cabal-install计算build plan的依据是Hackage目录，而这个目录在时刻更新，因此build plan结果是非确定性的，一个今天work的build也许明天不work；或者在build完另一个项目以后，不再work了。

为了解决cabal hell的问题，ghc和cabal-install的新版本都加入了一些workaround，比如ghc 7.10起支持import指明package和版本号、Package Database中版本号带上依赖Hash、cabal-install引入sandbox机制（类似Python的virtualenv，使得在愿意浪费空间的前提下hell问题可以说解决了）和模仿nix的new-build机制。

不过，FP Complete提出了另一种方案——既然cabal-install的dependency solving带来的非确定性是问题源头，那我们维护一个snapshot好了：这个snapshot是Hackage包集合的子集，为每个包指定了唯一的版本；每个snapshot发布时用CI build确保这些包在这些版本号下都能成功编译、构建文档和通过测试。当Hackage包发布新版本时，自动尝试将新版本加入下一个snapshot。这样一来，对于只依赖snapshot包的项目，编译时多个项目可以共享同一个snapshot的Package Database，速度快而且可靠、可重现，这对生产环境而言很重要。这就是Stackage项目的由来。Stackage snapshot有LTS和Nightly频道的区别，目前LTS major version为6，对应ghc 7.10.3，Nightly对应ghc 8.0.1，具体哪个snapshot包括了哪些包可以在Stackage官网查看。

作为Stackage项目的延伸，stack包管理器应运而生。用stack开发Haskell项目时，除了.cabal配置文件以外，需要另外维护一个stack.yaml配置文件，指定项目编译用的Stackage snapshot。而stack编译时不会像cabal一样维护全局的Package Database，而是每个snapshot对应自己的Package Database。如果不开发Cabal package，只是普通地通过stack调用ghc或ghci的话，stack会自动生成和使用一个“global project”的stack.yaml，这个global project的snapshot也是可以指定的。除了管理多个snapshot以外，stack可以自行安装和管理多个版本的ghc。需要用到不在snapshot中的Hackage包时，可以通过将其添加到stack.yaml的方式来“扩展”某个snapshot。

最后，对Windows下的Haskell开发者，需要提一下msys2。这是一个MinGW-w64的分支，移植了Arch Linux的pacman管理器，可以在Windows下使用Unix shell以及安装不少移植版的Unix库与工具。stack在windows下使用时，stack setup不仅会安装ghc，也会安装msys2，之后可以用stack exec执行pacman命令，安装某个Haskell包所用的Unix依赖，比如pkg-config/gtk/gtk3/glfw/sdl2，等等。不过如果本机已装过msys2，可以用--skip-msys选项跳过其安装，然后在全局的msys2的shell里使用stack。

- 我是不懂Haskell的吃瓜群众，需要使用pandoc、shellcheck、git-annex之类用Haskell写的工具——用平台自带的包管理器装。
- 我是Haskell小白，想要敲一些书上的代码，怎么装Haskell——装stack，然后用stack setup安装ghc，stack install安装所需库。
- 我是用cabal的小白，刚才装库装不上，好像是什么版本号冲突来着——执行rm -rf ~/.cabal && rm -rf ~/.ghc，然后参考上一条。

先敲这么多吧，这是先行版，讲了一些我认为最重要的工具链常识，希望能对Haskell初学者有所助益。如果有不明白的地方，欢迎在评论区提出，我会适量增补。另外这里只涉及build的问题，还有一些其他的有用开发工具——haddock（文档构建）、ghcid/intero/ghc-mod（编辑器插件）、hpc（code coverage），日后有空慢慢补上吧。

祝大家学习愉快！下期预告：带逛Hackage。

发布于 2016-07-31

Haskell 函数式编程 GHC（编程套件）

文章被以下专栏收录

魔鬼中的天使
主要会讨论关于函数式编程（haskell、scala）的内容。我会尽力讲的清晰明了，带你...

推荐阅读

也说Haskell

gua 发表于2gua的...

Haskell程序员的进化

新手fac n = if n == 0 then 1 else n * fac (n-1) Lisp程序员转行的fac = (\(n) -> (if ((==) n 0) then 1 else ((*) n (fac ((-) n 1))))) 学了模式匹配后fac 0 = 1 fac (n+1) =...

raino... 发表于Conti...

2018 Haskell 开发环境配置

最近尝试了下Idris，感觉开发环境（emacs）非常好，特别是case split的功能，可以把某个代数数据类型直接生成模式匹配展开式，还有proof-search，某些情况写出类型就能直接生成term，习惯了...

Marti... 发表于Indus...

Has

22 条评论

切换为时间排序

写下你的评论...

何幻 2 年前

仙境里的。。红尘里的。。666

赞

酿酿酿酿酿泉 2 年前

话说 stack setup 的下载体验，我这里简直想死，最后不知道怎么下载下来的...

赞

祖与占 2 年前

nix那套可以科普下?

1

邱焜 回复 祖与占 2 年前

nix是一个linux dist

赞 查看对话

祖与占 回复 邱焜 2 年前

nixos才是...

赞 查看对话

包遵信 2 年前

我觉得对每种语言，都有必要问它的支持者，你认为这门语言最大的坑是啥，不知道楼主能不能解答一下。

1

刘运峰 2 年前

接受科普的吃瓜群众表示，Haskell的module跟package和Python好像啊。😃

赞

大石匠 回复 包遵信 2 年前

作为断断续续搞了几年haskell的人谈点自己的体会吧
第一个 搞清楚自己为什么要玩这个，边玩边问，边问边玩，太多时候会有不玩了的想法。毕竟鲜有

第二是 准备好丢弃多年形成的编程习惯，毕竟没有赋值语句和类定义。尝试过在haskell下mock一个出来，烧脑。
第三 要较深入了解一个问题时，搜的资料多半会指向一堆的英文paper，还有人会扯上一堆数学，啥群论什么的。
第四 工具链没JAVA那么完善，没有完整的ide，非要有这个的就在折腾ide中退却了。不过如果已经习惯在非ide环境下折腾，这个不是大碍

搞haskell对我的体会是打开了另一个世界的窗户，如同碳基生物看到了硅基生物。妈的，程序还能这样。

有人说思想是自由的，而工具是其次。但是在程序设计领域看到的大多是后者禁锢了前者而不自知。所以用haskell来fuck下brain是很有益处的

9 查看对话


兔猫肉 回复 酿酿酿酿酿泉 2 年前

tuna有镜像

1 查看对话

李约瀚 回复 兔猫肉 2 年前

是啊，解决不少问题

赞 查看对话

李约瀚 2 年前

话说，有人科普一下 xmonad 和 ghc 的正确调试的姿势么?😂

赞


Jex Cheng 2 年前

stack不能不提package set啊：github.com/commercialha...

赞


祖与占 回复 兔猫肉 2 年前

build plan还是连到aws下...

1 查看对话

[已重置] 2 年前

stack的下载体验，砸电脑，生气

赞

赞

snapsisy 回复 大石匠 2 年前

brainfuck 跟haskell 有什么联系吗？以前听人家说是有的。。。

赞 查看对话

大石匠 回复 snapsisy 2 年前

本来不是说brainF*K语言，是在说操练下大脑的意思。
不过BF语言是基于图灵机的，而Haskell(FP)是基于lambda演算的。图灵机与lambda演算是两大不同的派系，都是为了解决可计算问题而建立的理论体系，但是这二者的计算能力是等价的。

赞 查看对话

张智城 1 年前

同问stack setup 下载体验慢，有用lantern还是慢

赞

cccc 回复 大石匠 1 年前

表示IDEA有插件支持Haskell，感觉挺好用的~

赞 查看对话

颜什么都不记得适 1 年前

stack setup 太慢根本没法用……换了 tuna 镜像也一样啊。放弃 Haskell 了= =

1