



## 使用 Cabal hook 构建复杂 Haskell 项目



Felis sap... 

函数式编程、编程语言、编程 话题的优秀回答者

已关注

rainoftime、考古学家千里冰封、圆角骑士魔理沙、酿酿酿酿酿泉、草莓大福等 28 人赞了该文章

首先说说“复杂”Haskell 项目是怎么回事。嗯，并不是下图这种意义上的复杂：





M...K... restructure directory

1 contributor

247 lines (191 sloc) | 7.37 KB

```
1  {-# LANGUAGE
2      MultiParamTypeClasses,
3      RankNTypes,
4      ScopedTypeVariables,
5      FlexibleInstances,
6      FlexibleContexts,
7      UndecidableInstances,
8      PolyKinds,
9      LambdaCase,
10     NoMonomorphismRestriction,
11     TypeFamilies,
12     LiberalTypeSynonyms,
13     FunctionalDependencies,
14     ExistentialQuantification,
15     InstanceSigs,
16     TupleSections,
17     ConstraintKinds,
18     DefaultSignatures,
19     UndecidableSuperClasses,
```



```
21 typeApplications;  
22 PartialTypeSignatures #-}
```

这里的复杂指的不是语言特性，而是 build process 的复杂性。对于 Haskell 初学者和普通用户而言，只要掌握一些基本的工具链常识（我在[@大魔头-诺铁](#)的专栏有简单讲过：[红尘里的 Haskell（之一）--Haskell 工具链科普 - 知乎专栏](#)），以及几个简单 stack/cabal 命令的用法，就可以创建和管理 Haskell 项目。

stack/cabal 之类工具的核心是 Cabal 库，它提供了描述 Haskell 的 package/module 元数据的格式，以及一些内建的 builder，在 configure/build/generate haddock/... 等流程中解析和处理这些元数据，并根据需要调用 GHC、linker、C 编译器等工具完成构建。按照 Cabal 文档的描述（[3. Package Concepts and Development](#)），提供一个 .cabal 文件，以及一个简单的 Setup.hs 脚本（对于大多数 package，对应的 Cabal build type 是默认的 Simple，这个脚本只有 2 行），stack/cabal 就可以自动帮你完成其他工作。

然而 Cabal 默认特性并不总是够用的。考虑以下 2 个例子：

- 实现 Haskell 绑定 C/C++ 库。这个库可能在系统中装了多个版本，需要一些特殊的逻辑去寻找和设定相关的 flag，甚至有可能需要把这个库的开发版打包到 Haskell 项目中自行构建和链接。
- 实现 Haskell 元编程，输出一堆 Haskell module 的源文件（也许还带着 haddock 文档）并将其作为 Cabal build target

对前一个例子，Cabal 本身提供了用 pkg-config 找配置，以及编译 Haskell 项目中的（不需要什么奇怪脚本预处理、不涉及 make 时吐出新代码的）C 源文件。然而不支持 C++，不支持编译动态生成新的 C/C++ 代码，也不能搞些奇怪的预处理（根据 Haskell 项目要求去 patch 原来库的代码），或者使用 ninja 之类的非主流 build 工具.....

对后一个例子，Haskell 元编程的标准做法是 Template Haskell，可以在编译期执行任意计算并生成各种声明。然而 Template Haskell 暂不能生成新 module，而且其语法树中并未考虑 haddock 文档的支持。

归根结底，我们需要在 Cabal 的 setup 程序里用 Haskell 实现自定义的 build 逻辑，比如调用奇怪的工具，或者魔改自己的 package 元数据。考虑到这样的需求，Cabal 提供了 hook 接口，可以在默认的 Simple builder 上面用自己的 callback 来取得数据并做一些微小的工作。

hook 的使用方法：首先将 your-package-name.cabal（或者如果你赶时髦用 hpack 的话，package.yaml）中的 `build-type` 一栏从 Simple 改为 Custom。另外，强烈推荐增加 `custom-setup` 信息，用于描述 Setup.hs 脚本自身的 Haskell 依赖，这些依赖与项目自身的各个 build target 的依赖是完全独立的，也不会共享默认编译选项、语言扩展等 flag。改完以后，就可以开始动手魔改 Setup.hs 脚本了。

和名字就可以看出它们分别在什么阶段会被触发，输入/输出信息的类型是什么。现在，支持用 Haskell 代码来自定 Haskell 项目的 build 逻辑以后，前面所说的问题也就可以解决了：比如需要 build C++ 库的话，可以在 preConf 或者 confHook 上面加上“调用 make，并将 .so/.a 复制到本 package 的默认 libdir”的逻辑；要是有点闲情逸致的话，甚至 make 也不必调，用 [Shake Build System](#) 把 build 逻辑自己重写一下也是完全 OK 的。至于 Haskell 代码生成，在生成代码并复制到源码目录中以后，暴力修改 LocalBuildInfo 中的 localPkgDescr，把私货夹带进去就可以开开心心地 build 啦。

当然，最糙快猛的做法，无疑是用其他语言写个脚本来生成和打包所有你需要的项目文件，这样做的话 SAN 值会掉 90%；或者善用 C 预处理器，在项目里到处 #include，然后像操作文本一样用宏吐出一堆 Haskell 代码，这样做 SAN 值会掉 50%。不过，最理想的做法，当然是——所有在其他系统里用到的奇技淫巧，都能用 Haskell 波澜不惊地实现出来。Best magic is no magic。

考虑到 Cabal 的 hook API 并不是一个非常 well-documented 的东西，而且行为在 stack/cabal-install 下表现还不完全一致，初学者为了避免一些陷阱，也许需要跟踪整个 Cabal build process，尤其是偷看这些 hook 在什么时机被触发、传入的参数和计算结果分别是什么。所以我写了一个简单的打日志的工具：[TerrorJack/Cabal-playground](#)。这个项目包含 foreign C code/library/executable/test-suite/benchmark 各一个，以及一个用于详细观测 Cabal build process 的 Setup.hs 脚本（比原本最 verbose 的选项更 verbose），只要设定一个环境变量并启动 build，就可以从日志读到每一个 hook 的触发时机以及调用参数/结果。这个 Setup.hs 脚本很容易通过简单修改迁移到其他的 Haskell 项目，用于 debug 其他项目的构建流程。

P.S. 为什么想到写这玩意呢？嘛，我最近写的 Haskell 代码和开头的图一比，画风完全不一样。。

```
707  foreign import ccall interruptible "RelooperAddBranchForSwitch"
708          c_RelooperAddBranchForSwitch ::
709          RelooperBlockRef ->
710          RelooperBlockRef ->
711          Ptr BinaryenIndex ->
712          BinaryenIndex -> BinaryenExpressionRef -> IO ()
713
714  foreign import ccall interruptible "RelooperRenderAndDispose"
715          c_RelooperRenderAndDispose ::
716          RelooperRef ->
717          RelooperBlockRef ->
718          BinaryenIndex -> BinaryenModuleRef -> IO BinaryenExpressionRef
719
720  foreign import ccall interruptible "BinaryenSetAPITracing"
721          c_BinaryenSetAPITracing :: CInt -> IO ()
```



```
824         , [ "ast_utils.h"
825             , "compiler-support.h"
826             , "literal.h"
827             , "mixed_arena.h"
828             , "pass.h"
829             , "wasm-builder.h"
830             , "wasm-traversal.h"
831             , "wasm-type.h"
832             , "wasm.h"
833             , "emscripten-optimizer" </> "istring.h"
834             , "support" </> "name.h"
835             , "support" </> "threads.h"
836             , "support" </> "utilities.h"
837         ])
838     ]
839
840     main :: IO ()
841     main =
842         defaultMainWithHooks
843         simpleUserHooks
844         { confHook =
845             \t c ->
846                 let Distribution.Simple.Setup.Flag p =
847                     libdir $ configInstallDirs c
848                 in do buildBinaryen (fromPathTemplate p)
849                     confHook
850                     simpleUserHooks
851                     t
852                     c
853                     { configExtraLibDirs =
854                         fromPathTemplate p : configExtraLibDirs c
855                     }
856         }
```

其中大多数 Haskell 代码是我用 Python 脚本吐出来的。不能用 Haskell 完成整个 workflow 的每一个环节让我掉了许多 SAN 值。所以有了这篇专栏。

编辑于 2017-03-30




赞赏

还没有人赞赏，快来当第一个赞赏的人吧！

- Haskell
- 函数式编程
- GHC (编程套件)


赞同 28 11 条评论 分享 收藏 ...

文章被以下专栏收录

- 

不动点高校现充部

一切与编程语言理论、函数式编程相关的杂谈。

已关注
- 

雾雨魔法店


<http://zhuanlan.zhihu.com/marisa/20419321>

已关注


推荐阅读

- 


haskell开发环境配置

丁光寸阴 发表于Haske...
- 

幻想中的Haskell - Compiling Combinator

圆角骑士魔... 发表于雾雨魔法店
- 

在Haskell中模拟dependent type

mirone
- 

剖析

祖与

11 条评论 切换为时间排序

写下你的评论...



圆角骑士魔理沙

有必要打码吗。。。另：投稿！

1 年前



打码的MK有着别样的色气 (

👍 3    💬 查看对话



圆角骑士魔理沙 回复 Felis sapiens (作者)

1 年前

60%的纯爱党表示sad

👍 赞    💬 查看对话



祖与占

1 年前

用 Shake 不好吗 (

👍 赞



Felis sapiens (作者) 回复 祖与占

1 年前

build Haskell 的部分用 shake 优势不大，因为主要的重活是ghc -make干，自己做 dependency tracking 然后用 ghc single shot + shake 并行化的话效率不行；至于build non-Haskell 的部分，不是不好，不是小好，是大好

👍 赞    💬 查看对话



祖与占 回复 Felis sapiens (作者)

1 年前

啊 我就是说类似 build Haskell 之前一些预备工作的部分...

👍 赞    💬 查看对话



Yutong Zhang

1 年前

DDF的这个列表已经变得这么长了啊.....

👍 1



梨梨喵

1 年前

这打码2333

👍 赞



李约瀚

1 年前

大家的注意力都被打码的东西吸引了

👍 赞



酿酿酿酿酿泉

1 年前

有种援力!

👍 赞





所以什么叫san值啊？

赞

