



关于 Haskell 开发环境的二三事



Felis sap... 

函数式编程、编程语言、编程 话题的优秀回答者

已关注

圆角骑士魔理沙、祖与占、刘雨培、霜月琉璃、酿酿酿酿泉等 59 人赞了该文章

本文不定期更新，汇总一些与 Haskell 开发环境相关的 trick。

使用 Hackage/Stackage 镜像

见 [Stackage 镜像使用说明 - 知乎专栏](#)。

使用全局 msys2

在 Windows 下，GHC 分发的二进制包自带一套（不完整的）mingw-w64 工具链，GHC 默认会在自己的安装目录下寻找 mingw-w64 的 gcc（而不是在 %PATH% 下寻找）。

而 stack 除了 GHC 以外，默认还会夹带安装 msys2，所有的 stack 命令都会在这个 msys2 的 mingw64 shell 中执行，这样一来，虽然 gcc 用的还是 GHC 自带的版本，不过在链接第三方库、使用一些 shell 工具（比如 network 之类需要运行 Unix-style configure script 的库）时，会使用 msys2 提供的库和工具。

msys2 是自带版本管理器（pacman）和终端模拟器（mintty）的，如果除了 Haskell 开发以外并不在其他地方用到的话，使用 stack 默认安装的版本即可（可以用 stack exec bash 启动终端）。不过如果你和我一样用 Windows 开发而且：

- 有强迫症（不喜欢多个 gcc.exe 待在系统里，尤其是版本号不够新的）
- 在 C/C++ 开发中也有用到 msys2 来装依赖库



- 在 %APPDATA%\stack\config.yaml 中添加 skip-msys: true 一行。stack 会跳过 msys 检测和安装的环节，不过在需要链接 msys2 中的库或调用其工具时，不能在普通的 cmd/PowerShell prompt 下运行 stack 命令，而要在 msys2 的 mingw64 shell 下运行。单是改 %PATH% 是不行的
- 用 mklink /D 命令创建一个符号链接，从 C:\msys64 指向 %LOCALAPPDATA%\Programs\stack\x86_64-windows\msys2-20161025，同一目录下新建 msys2-20161025.installed 文件，内容为 installed（无换行符）。这样 stack 会自动把全局的 msys2 当成自家的调用，也就可以在普通的 cmd/PowerShell prompt 下运行所有 stack 命令了。对了，ghc 安装目录下的 mingw 目录也可以删掉然后换成符号链接

关于编辑器插件

对 Atom/VSCode 用户：很遗憾，ghc-mod 仍然是相对最好用的工具。ghc-mod 可以通过 stack/cabal-install 安装，注意不同 ghc 版本需要分开 build。build 完以后保证 %PATH% 里能找到，相应编辑器插件的类型检查和补全功能就 work 了。

对 Emacs 用户：不妨用 intero，而且也有对应的插件在 MELPA 上可以安装。

对 Vim 用户：咱不用 Vim 所以也不知道该写啥（

对 Notepad 用户：代码补全肯定没法指望了，不过 type check on save 还是没问题的。可以单开一个命令行窗口，然后运行 ghcid 或者 stack build --fast --file-watch，每次代码保存的时候会触发编译，可以从命令行读到警告和错误。

hlint 和 hindent 也推荐用一下。前者能够及时发现代码里比较低级的 anti-pattern，后者能够格式化代码让它看上去更像 Chris Done 写出来的。嗯，Chris Done 的风格总体还是比较能看的。

使用 stack 运行 cabal-install

这年头虽然 cool kids 更喜欢用 stack，cabal-install 偶尔还是用得上的——比如需要安装一些没有被 Stackage 收录的工具，又实在懒得去改 stack.yaml 把依赖库写进去的场合。cabal-install 通过 \$PATH 来寻找 ghc，我们可以通过 stack exec 来帮 cabal-install 使用 stack 安装和管理的 ghc，像这样：stack exec cabal -- exec cabal -- install purescript。别忘了用 cabal sandbox。

用 WSL 配置 Linux 环境



- 用到了不支持 Windows 平台的 Hackage 包（一般来说原因有二：transitive dependency 里出现了 unix；本来支持 Windows 但是 linker flag 写错了，发 pull request 作者不理你）
- 想要确保跨平台性（其实也就跨 x64 Windows/Linux 两个平台，别的懒得管），不想等 Travis CI / AppVeyor 慢吞吞 build 出来，不想扔自家 VPS 上 build（钱少内存少，一 swap 就慢出翔），也不想开虚拟机（给 Chrome 和正在推的 galgame 多省点内存用）

所幸我们有 WSL。WSL 从 Build 14986 开始能够运行 GHC，包括 stack 自己安装的以及通过 ppa 安装的版本。在 WSL 下也安装 stack，然后在 ~/.stack/config.yaml 中加上 allow-different-user: true（否则会有权限问题），通过 /mnt/.. 定位到 Windows 文件系统里的项目目录，即可和 Windows 双平台同时编译运行，彼此不会冲突。在编辑器里单开一个 Terminal 然后启动 WSL bash，运行 ghcid 或者 stack build --fast --file-watch-poll，可以收获双份的编译错误哦。

WSL 下运行 GHC 的一个问题是暂时不能运行 GHC 8，因为 GHC 8 的运行时内存管理会 mmap 许多内存，WSL 尚不能很好地处理，速度极慢，具体见 [`stack ghc` painfully slow · Issue #1671 · Microsoft/BashOnWindows](#)。这个 issue 的解决需要等 Creators Update 以后。GHC 自身的 build system 提供了一个 configure flag 可以禁用 GHC 8 的新内存管理器，所以在 WSL fix 这个 issue 之前，我们可以手动编译和安装 GHC（详细参考 [Building - GHC](#)）：

```
stack config set resolver lts-6
stack --install-ghc install alex happy
sudo apt install git autoconf automake libtool make libgmp-dev ncurses-dev libffi-dev
git clone --recursive https://git.haskell.org/ghc.git
cd ghc
./boot
./configure --prefix ~/.local --with-ghc=$HOME/.stack/programs/x86_64-linux/ghc-7.10.3
make -j8
sudo make install
```

WSL 的另一个作用是支持 stack 的 Nix/Docker 集成。Nix 现在可以直接使用官方的安装脚本安装了，并不需要额外的 hack。而 Docker 特殊一些，WSL 并没有提供虚拟化相关的接口，所以 Docker daemon 是跑不起来的，但是我们可以安装 Docker for Windows 作为后端（底层基于 Hyper-V 上的 Alpine Linux），然后在 WSL 环境下连接这个后端：

```
sudo apt install apt-transport-https ca-certificates curl software-properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo apt add-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu xen
sudo apt update
sudo apt install docker-ce
export DOCKER_HOST=tcp://127.0.0.1:2375
```



在 ghci 中加载和使用 object code

在项目目录里使用 `stack/cabal-install` 来 build 时，GHC 会将 Haskell 源代码通过 native code generator 生成 object file 并进行链接。而启动 ghci 时，ghci 并不会使用编译和链接好的 native code，它会重新将 Haskell 源代码编译一遍，生成 ghci 专用的 bytecode，这些 bytecode 解释执行，并不生成 native code。

在启动 ghci 时，使用 `-fobject-code` 选项（stack 的对应命令：`stack ghci --ghci-options -fobject-code`），可以使 ghci 直接载入预先 build 好的 object code，能够节省启动 ghci 时重新编译项目所有 Haskell 代码的时间（尤其是用到 Template Haskell，需要编译两次）。

另外，这个选项能够严格保证解释模式下的代码用到项目中的函数时，该函数的行为严格与编译模式一致。虽然两个模式共用一个前端，支持的语言特性一样，然而细微的差别是存在的：解释模式下的自定义重写规则不会发动，而一些函数时间/空间性能的保证依赖于重写规则。这样可能出现 ghci 下莫名其妙爆内存，然而编译后问题消失的现象。

ghci 下使用 object code 的缺点是无法使用 ghci 内置的断点调试功能。考虑到咱们是 Haskell 开发者——谁还记得上次打断点是哪年哪月呢（

在 ghci 中使用自定义 pretty-printer

ghci 有一个选项 `-interactive-print`，允许指定一个表达式作为 custom printer，表达式类型为 `C a => a -> IO ()`。我们可以使用 `pretty-show` 包和 `hscolour` 包实现一个带高亮的 pretty-printer 替换掉默认实现：

```
import Text.Show.Pretty (ppShow)
import Language.Haskell.HsColour (Output(TTY), hscolour)
import Language.Haskell.HsColour.Colourise (defaultColourPrefs)
let pPrint = Prelude.putStrLn . hscolour TTY defaultColourPrefs False False "" False .
:set -interactive-print pPrint
```

现在可以实验一下效果：

```
:{
data BinTree a = Nil | Cons a (BinTree a) (BinTree a) deriving (Show)

f :: Word -> BinTree Word
f 0 = Nil
f n = Cons n t t where t = f (n-1)
:}
```



编辑于 2017-04-11

「真诚赞赏，手留余香」

赞赏

还没有人赞赏，快来当第一个赞赏的人吧！

Haskell

GHC（编程套件）

函数式编程

▲ 赞同 59



● 20 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

推荐阅读



haskell开发环境配置

寸光阴

发表于Haske...



幻想中的Haskell - Compiling Combinator

圆角骑士魔...

发表于雾雨魔法店

学 Haskell 果然是要趁早

几年前，length 的类型是介个样子
`length :: [a] -> Int` 现在你打开 GHCi 捏，是介个样子滴，（搜 FTP，并没有真相 `length :: Foldable t => t a -> Int` 当然了，这其实并不会...

Cosmia Fu



剖析

祖与

20 条评论

⇌ 切换为时间排序

写下你的评论...



其实 vscode 也有基于 intero 的 haskero。

marketplace.visualstudio.com... 这里还有大量其他的东西。

还有一个 ucl 开发的 haskelly, 和 haskero 好像没很大区别, 可惜目前 (4 JUN 2017) 不支持 windows, 因为 stack-run 需要 unix 这个 package。

这些 package 的安装方法都是需要 cabal 安装其他可执行文件的。相对比较复杂, 不是在 vscode 里面点点就行了。。。

 赞



Felis sapiens (作者) 回复 hhhhhhhhhh

1 年前

我试过, 不好用

 赞  查看对话



hhhhhhhhh 回复 Felis sapiens (作者)

1 年前

啊, 这样, 主要是 haskero 好歹有那么一点点 intellsense, ghc-mod 好像没有。我只是初学, 还没太用过里面的很多 feature。

 赞  查看对话



Felis sapiens (作者) 回复 hhhhhhhhhh

1 年前

Atom 的 ide-haskell 是有补全的, VSCode 没有。

 赞  查看对话



圆角骑士魔理沙

1 年前

司机司机投稿不0 0

 赞



Felis sapiens (作者) 回复 圆角骑士魔理沙

1 年前

这篇和上篇不投了, 不那么好玩。等我写完《我们不用很麻烦很累就可以搞硬件开发》再投你专栏

 3  查看对话



NonExist

1 年前

Vim 我以前用 ghc-mod, 速度太慢了。现在就 stack ghci 里面开 vim ...

 赞



圆角骑士魔理沙 回复 Felis sapiens (作者)

1 年前

吼啊, 随你, 揉揉司机!!

 赞  查看对话



推荐用不带gcc的ghc: github.com/Alexpux/MINGW...。最早是我改的。

👍 赞



风干鸡 回复 Felis sapiens (作者)

1 年前

装了 haskelly 插件就有了

👍 赞

💬 查看对话



木木兽 回复 Chen Aaron

1 年前

这个链接好像失效了?

👍 赞

💬 查看对话



凉宫礼

1 年前

vim 有 haskell vim now, github 上的, 一键式傻瓜配置

👍 赞



nameoverflow

1 年前

邵菊苣威武..... GHC 8.0.2 在 win10 的新 build 上有 BUG, gcc 挂得一塌糊涂, 手动换了之后终于好了

👍 赞



熊森特 回复 Felis sapiens (作者)

1 年前

是要介绍 clash 嘛?

👍 赞

💬 查看对话



Felis sapiens (作者) 回复 熊森特

1 年前

正是。我可是买了板子的

👍 赞

💬 查看对话



Fallenwood

1 年前

我把ghc的mingw干掉换成了msys2的mingw64应该不会出问题吧。。

👍 赞



Felis sapiens (作者) 回复 Fallenwood

1 年前

我就这么干的, 稳

👍 赞

💬 查看对话



PSR-0E9H 回复 Felis sapiens (作者)

1 年前





李约瀚

1 年前

mklink ...这个东西 win10 不自带了。。。



赞



李约瀚 回复 Felis sapiens (作者)

1 年前

坐等

