Prof. Dr. Peter Thiemann

Janek Spaderna
janek.spaderna@pluto.uni-freiburg.de

Winter Term 2022/23
Discussion on 2022-10-21

Functional Programming
http://proglang.informatik.uni-freiburg.de/teaching/functional-programming/2022/

Installation and Setup

Overview

These instructions will guide you through the installation process of the following tools:

GHCup is a version manager for the Haskell toolchain. The three tools listed below will be installed through GHCup.

Although package managers usually provide *a* version of GHC, it is often outdated and you will run into trouble sooner rather than later.

GHC is the Haskell compiler.

cabal is the Haskell package manager.

Haskell Language Server (abbreviated as HLS) is the language server for Haskell. It provides completion, inline errors, code actions etc. in supporting editors (e. g. VSCode). *This piece is optional.*

Additionally we will describe setup of the VSCode Haskell extension. It makes use of the Haskell Language Server. Other editors may be used as well but please refer to the documentation of HLS for further information.

Installing GHCup

Follow the installation instructions at `https://www.haskell.org/ghcup`. The process should be mostly self-explanatory. Make sure to read the messages printed to the screen carefully. Below we provide some additional pointers and suggestions which answers to provide to the prompts during the installation.

*nix Systems

If the command fails immediately you might have to install `curl` first, e. g. by issuing `apt install curl` or similar.

Suggested answers:

- **p**repend the binary location to `PATH`.
- **y**es, install the Haskell Language Server.
- **n**o, do not improve integration with *stack*.

Windows

The official Getting Started instructions contain a video showing the installation process.

The default choices mostly suffice except for

- **y**es, install the Haskell Language Server.
- **n**o, do not install *stack*.

If you do not care for the language server feel free to answer “no” at the corresponding prompt. Manual installation at a later point in time is possible as well.

The installation will then proceed to download the actual GHCup executable. On *nix Systems you will have to verify that the listed system dependencies are installed before downloading GHC, *cabal,* HLS, and finally adjusting your `PATH` variable.

Verifying the Installation

If the installation completed as planned you should be able to issue `ghcup list` to get a listing of available and installed components. On *nix systems `ghcup tui` provides a (poor man's) graphical user interface. Remember to execute these commands in a new shell session.

Verify that the output of `ghc --version` and `cabal --version` matches the versions shown by GHCup. Failed commands and/or mismatching output suggest a problem in setting the `PATH` variable. You should also be able to launch GHCi sessions with the command `ghci`.

A First Project

Create and enter a new directory, e. g. `hello-haskell`. Inside, run `cabal init` which will create a basic project structure. By specifying the flag `-i` or `--interactive` you can get an interactive experience.

Enter the new directory and take a look at the files:

`hello-haskell.cabal` The name of this file defaults to the parent directory. This so called "cabal-file" contains a description of the project, its dependencies and lists its source files. *cabal* needs this information to derive the correct GHC invocations.

`app/Main.hs` This is the executable's entry point, the `Main` module. Here you can see the `main` function.

By executing `cabal run` from inside the project directory you can build and run your executable. After some output regarding the build you should see `Hello, Haskell!` printed to the screen. Afterwards you will be able to see an additional directory called `dist-newstyle` which contains the build products and intermediate files.

The command `cabal repl` will drop you into a GHCi session in the context of the project. This means dependencies are available and all source files will be loaded. If you are working in the context of a project it is always preferable to launch GHCi sessions via `cabal repl` compared to bare `ghci` invocations.

VSCode Setup

VSCode can be downloaded from `https://code.visualstudio.com`. Then install the Haskell extension.

When you open your first Haskell file, let's say `Main.hs` from the previous section, the extension will ask you how it should manage/discover the Haskell Language Server. Choose "Automatically via GHCup."

For a quick example, remove the type signature (that is all of line 3). In the beginning it may take a few seconds to startup but eventually you should see the type signature re-appear in small gray text. Clicking on said text will insert the type signature. Removing the `putStrLn` will give you an example how type errors are presented.

Note: (but speaking only from one experience) on a Windows system you might have to close and reopen VSCode a few times for everything to get settled. As soon as you can follow along with the steps described in the paragraph above, everything works as expected.