Haskell on CentOS 7

Jorge Nunes

November 4, 2016

Abstract

Where we summarily describe how to install the usual Haskell development tools on CentOS 7, and gleefully proceed to compile and run the canonical "hello, world" program.

1 Introduction

This story begins with us wanting to write Haskell programs.

Functional programming is in the air. From C++ to Java, from Python to Javascript, all your parents usual programming languages have for a while been getting pimped up with functional programming like features. More recent mainstream languages, like Scala or Rust, already addressed functional programming head-on from the start. Of course your grandparents and great-grandparents were also doing it back in the day. So functional programming seems to be one of those good ideas that just take a while to get spread around. Among all the functional programming brouhaha Haskell seems to be getting some increased mind share as of late (at the time we type these words, on the second half of the second decade of the 21st century AD). So let us assume that all the verbiage we have been spouting is by now enough justification for us to want to know more about Haskell.

The physical act of writing Haskell programs, or for that matter programs in any other language, does not seem to be that difficult. You can do it in your own head. Or by writing it on paper. Or even write it using a computer and your text editor of choice (which, unless you are using `cat > myfile.txt` to edit your files, will be Emacs, the *One True Editor*™).

And if writing Haskell programs was all that we wanted our story would then be ended right here, and right now. But we do want more than that. Not only do we want to write Haskell programs, we also want to run those programs! Yes, indeed we are daring and audacious.

Now running Haskell programs, that is a tad little more challenging. Here we will just have to rely on common wisdom regarding how we are to proceed. In order to run Haskell programs we will use a compiler to compile the Haskell source code into an executable. It is that executable that can then be executed (if you pardon the pun), meaning we will then be running the Haskell program.

And that, dear reader — please do allow us to break the fourth wall, and to address you as such — will be the gist of our story. We will endeavor to tell the tale on how to get ourselves an Haskell compiler, and how to compile and run an Haskell program. Humble as this goal may be, we hope it will be a small contribution to keeping us — and you, esteemed reader — on the path to being a better person.

The discerning reader — that would be you, yes, gracious reader — is surely by now wondering why not just ask Google about all this. And oh so right would be that reader of ours. Indeed all the useful information one will get out of this text was procured through the Google search engine. But we want to believe we added value by collating all that information, and presenting it in a clear, straightforward, no-frills way.

Let us continue in earnest with out story. In the following sections we will address three very specific subjects.

- Installing the Haskell development tools on CentOS 7 — These include the Haskell compiler, the magical tool for converting an Haskell source file into an executable.
- Writing and compiling the Haskell "hello, world" program — Here we confirm the Haskell compiler has been installed, and is working as intended.
- Enabling the Emacs mode for Haskell — Your favorite editor (that would be Emacs, yes, we know) obviously includes a mode for editing Haskell source files. Here we describe how to make the Haskell mode available in Emacs (which we do know, we are very mindful of that, is your favorite text editor).

Without further ado let us then continue.

2 Installing Haskell development tools on CentOS 7

The Haskell development tools are available from the EPEL repository provided by the Fedora Project.

So as very first step we need to configure Yum to include the EPEL repository in the set of repositories used for installing packages. Unless, of course, we had already done it in some distant — or perhaps not — past, being the case that EPEL offers so many usefull packages.

Adding the EPEL repository to Yum has fortunately been made simple by the good people at the Fedora Project. There is a CentOS `epel-release` package with the express intent of making it simple to configure Yum to use the EPEL repository.

So let us then install said `epel-release` package. Like this:

```
[root@localhost ~]# yum install -y epel-release
```

After the command above completes successfully, the EPEL repository is now part of the set of repositories that Yum uses for fetching packages. And we can now happilly proceed with installing the Haskell development tools.

```
[root@localhost ~]# yum install -y haskell-platform
```

Well, that was easy. Let us do a quick check.

```
jfn@localhost:~$ ghc --version
The Glorious Glasgow Haskell Compilation System, version 7.6.3
jfn@localhost:~$ ghci
GHCi, version 7.6.3: http://www.haskell.org/ghc/  :? for help
Loading package ghc-prim ... linking ... done.
```

```
Loading package integer-gmp ... linking ... done.
Loading package base ... linking ... done.
Prelude>
```

All is cool! Hooray! We did it!

3 Haskell hello world

We now have the Haskell compiler installed. This means we are ready to create our very first Haskell program. Following the time honored tradition long put in place by our elders, our very first program in a programming language will have to be the canonical "hello, world".

We start with creating the file with the source code. That is easily done using everyone's favorite text editor (that would be Emacs by the way). The source code is shown below. We could have named that file whatever grandiose name we wanted, but having it named `hello.hs` just sort of feels right.

```
main = do
  putStrLn "Hello, world!"
```

With our Haskell source file ready, we need to compile it to generate the executable binary. Let us just do it.

```
jfn@localhost:~$ ghc --make hello.hs
[1 of 1] Compiling Main             ( hello.hs, hello.o )
Linking hello ...
```

Finally we have the executable. All that is left is to run it. And run it we will!

```
jfn@localhost:~$ ./hello
Hello, world!
```

Wow! Just wow! And with this, gentle reader, we have surely reached one of the high points in this most unpretentious story.

4 Emacs mode for Haskell

There is an Emacs mode for editing Haskell source code. This Emacs mode provides syntax highlighting and auto-indentation.

A CentOS package is already available that includes this Emacs mode. To install it:

```
[root@localhost ~]# yum install -y emacs-haskell-mode
```

After installing the above package the Emacs mode works right out of the box. The Haskell mode is automatically activated when opening a file with a `.hs` extension. Still, one thing you almost surely will want to do is enable the automatic indentation for this mode. To enable automatic indentation just add to the `custom-set-variables` on your `.emacs` file the following parameter:

```
(custom-set-variables
 '(haskell-mode-hook '(turn-on-haskell-indentation)))
```

5 Epilogue

And that's it! We are happy campers! Finally ready to tread the flower covered golden path of the Haskell way on our journey to the fabled programming nirvana.