Felix Documentation Master

Release 2016.07.12-rc1

Aug 01, 2020

Contents

THIS IS A WORK IN PROGRESS!

Contents:

CHAPTER 1

Felix documentation Master

This is the master document for Felix documentation. It provides some general description of Felix, along with links to more specific documentation.

1.1 Specific Documents

- Documentation Master (this document) <http://felix-documentation-master.readthedocs.io/en/latest/>
- Installation and Tools Guide <http://felix-tools.readthedocs.io/en/latest/>
- Felix Language Reference Manual <http://felix.readthedocs.io/en/latest/>
- Felix Tutorial <http://felix-tutorial.readthedocs.io/en/latest/>
- Felix Library Packages <http://felix-library-packages.readthedocs.io/en/latest/>
- Articles on Modern Computing <http://modern-computing.readthedocs.io/en/latest/>
- Felix Home Page <http://felix-lang.github.io/felix/>
- Git Repository <https://github.com/felix-lang/felix>

1.2 General Description

Felix is a high level statically typed programming language designed with several key features in mind.

1.2.1 lightspeed performance

Which means, as fast as C, if not faster

1.2.2 C and C++ ABI compatibility

the ability to embed exising C and C++ code

1.2.3 Ease of use

as easy to use as a scripting language‘, which means no make files or switches for basic operation

1.2.4 high reliability

which means a fully statically typed language, for which reasoning about correctness is well supported

1.2.5 programmers toolkit

which means we provide many useful features and libraries, with multiple ways to combine and use them according to the application requirements and programmers taste

1.2.6 flexible deployment

which means the system can be used both as a personal development system, as well as for enterprise level team projects

1.2.7 write once run anywhere

the same code working the same way on all platforms

1.3 Language Design Goals

The Felix language has a number of important design goals.

- full integration of *coroutines* as core control structures
- full support for functional programming including # parametric polymorphism, # Haskell style type classes # a wide range of type constructors including
  - tuples
  - arrays
  - records
  - structs
  - anonymous sums
  - traditional nominally typed variants
  - generalised algebraic types (GADTs)
  - polymorphic variants
  - subtyping
  - uniqueness types

  - row polymorphism for records
  - first class functions and procedures
  - pointers
  - first class projections and injections
- expanded products: no boxing
- garbage collection
- algol like imperative programming as a subset of the coroutine system
- Java like objects and interfaces
- dynamically loadable plugins
- asynchronous I/O support
- pre-emptive threading support
- user defined grammar
- LaTeX/AMSTeX symbol set

CHAPTER 2

Indices and tables

- genindex
- modindex
- search