Eolian

Automatic EFL binding generation and more

Daniel Kolesa
Samsung Open Source Group
d.kolesa@osg.samsung.com
@octaforge
FOSDEM 2016

Introduction

What are we dealing with here?

What is EFL?

What is EFL?

- Enlightenment Foundation Libraries

What is EFL?

- Enlightenment Foundation Libraries
- A suite of graphics and other libraries (UI toolkit etc.)

What is EFL?

- Enlightenment Foundation Libraries
- A suite of graphics and other libraries (UI toolkit etc.)
- Originally created for the Enlightenment desktop shell

What is EFL?

- Enlightenment Foundation Libraries
- A suite of graphics and other libraries (UI toolkit etc.)
- Originally created for the Enlightenment desktop shell
- Cross platform

What is EFL?

- Enlightenment Foundation Libraries
- A suite of graphics and other libraries (UI toolkit etc.)
- Originally created for the Enlightenment desktop shell
- Cross platform
- Significant usage includes the Tizen operating system

The problem

The problem

- We provide a C API

The problem

- We provide a C API
- We need to use the API from different languages

The problem

- We provide a C API
- We need to use the API from different languages
- Maintaining bindings manually is difficult

The problem

- We provide a C API
- We need to use the API from different languages
- Maintaining bindings manually is difficult
- Can we generate them?

The problem

- We provide a C API
- We need to use the API from different languages
- Maintaining bindings manually is difficult
- Can we generate them?
- With the right tooling, yes we can

What is Eolian?

What is Eolian?

- It's several things

What is Eolian?

- It's several things
- It's a declarative format for describing APIs

What is Eolian?

- It's several things
- It's a declarative format for describing APIs
- It's a C library to deal with these declarations

What is Eolian?

- It's several things
- It's a declarative format for describing APIs
- It's a C library to deal with these declarations
- It's a generator for the core C API

Why is Eolian useful?

Why is Eolian useful?

- Language independent API descriptions

Why is Eolian useful?

- Language independent API descriptions
- Automatic generation of bindings for any language

Why is Eolian useful?

- Language independent API descriptions
- Automatic generation of bindings for any language
- Improved documentation

Why is Eolian useful?

- Language independent API descriptions
- Automatic generation of bindings for any language
- Improved documentation
- Better tooling

Why is Eolian useful?

- Language independent API descriptions
- Automatic generation of bindings for any language
- Improved documentation
- Better tooling
- The possibilities are endless

Former state

EFL before Eo

How was it?

How was it?

- Normal C API

How was it?

- Normal C API
- A lot of duplicated functions

How was it?

- Normal C API
- A lot of duplicated functions
- Difficult to bind

How was it?

- Normal C API
- A lot of duplicated functions
- Difficult to bind
- Existing bindings often out of date

Eo

Eo

- We decided for an object system

Eo

- We decided for an object system
- Goal: preserve C API and legacy compatibility

Eo

- We decided for an object system
- Goal: preserve C API and legacy compatibility
- Goal: preserve API/ABI compatibility even when adding methods

Eo

- We decided for an object system
- Goal: preserve C API and legacy compatibility
- Goal: preserve API/ABI compatibility even when adding methods
- Existing solutions all had drawbacks

Eo

- We decided for an object system
- Goal: preserve C API and legacy compatibility
- Goal: preserve API/ABI compatibility even when adding methods
- Existing solutions all had drawbacks
- Therefore Eo was created

Eo

Eo

- Eo is an object system written in C

Eo

- Eo is an object system written in C
- Provides inheritance, interfaces, mixins, etc.

Eo

- Eo is an object system written in C
- Provides inheritance, interfaces, mixins, etc.
- Provides API/ABI compatibility and easy legacy wrappers

Eo

- Eo is an object system written in C
- Provides inheritance, interfaces, mixins, etc.
- Provides API/ABI compatibility and easy legacy wrappers
- But Eo itself is not enough

Eo

- Eo is an object system written in C
- Provides inheritance, interfaces, mixins, etc.
- Provides API/ABI compatibility and easy legacy wrappers
- But Eo itself is not enough
- A way to describe Eo classes was necessary

Eolian

Eolian

- Thus Eolian was born

Eolian

- Thus Eolian was born
- We can describe all Eo objects effortlessly

Eolian

- Thus Eolian was born
- We can describe all Eo objects effortlessly
- We can use these descriptions to generate bindings or C APIs

Eolian

- Thus Eolian was born
- We can describe all Eo objects effortlessly
- We can use these descriptions to generate bindings or C APIs
- We can also use them in tooling

SAMSUNG
Open Source Group

Eolian

The basics

Eo file structure

Eolian library

Eolian library

- We provide C API to deal with Eo files

Eolian library

- We provide C API to deal with Eo files
- Does parsing, memory management and utilities

Eolian library

- We provide C API to deal with Eo files
- Does parsing, memory management and utilities
- A fully compliant reference parser

Eolian library

- We provide C API to deal with Eo files
- Does parsing, memory management and utilities
- A fully compliant reference parser
- Can be bound to other languages

Generators

Generators

- Written using the provided APIs

Generators

- Written using the provided APIs
- They emit the necessary glue code

Generators

- Written using the provided APIs
- They emit the necessary glue code
- Can be done several ways depending on the language

Generators

- Written using the provided APIs
- They emit the necessary glue code
- Can be done several ways depending on the language
- EFL has core generators for C, C++ and Lua

Generators

- Written using the provided APIs
- They emit the necessary glue code
- Can be done several ways depending on the language
- EFL has core generators for C, C++ and Lua
- The C generator is the actual C API of EFL

C generator

C generator

- We generate our own API

C generator

- We generate our own API
- Reduces maintenance overhead (only needs Eo files)

C generator

- We generate our own API
- Reduces maintenance overhead (only needs Eo files)
- Helps ensure correctness of our Eolian implementation

C generator

- We generate our own API
- Reduces maintenance overhead (only needs Eo files)
- Helps ensure correctness of our Eolian implementation
- Provides a reference for other generators

Generated C code

Other generators

What else do we get?

Non-binding tooling

Non-binding tooling

- Eo files can be used for further analysis

Non-binding tooling

- Eo files can be used for further analysis
- Example: GUI builder

Non-binding tooling

- Eo files can be used for further analysis
- Example: GUI builder
- Widgets as Eo classes, app doesn't need to know about them

Non-binding tooling

- Eo files can be used for further analysis
- Example: GUI builder
- Widgets as Eo classes, app doesn't need to know about them
- Example: documentation generator

Non-binding tooling

- Eo files can be used for further analysis
- Example: GUI builder
- Widgets as Eo classes, app doesn't need to know about them
- Example: documentation generator
- Generate documentation for APIs in different formats


SAMSUNG
Open Source Group

- A core generator in the EFL

- A core generator in the EFL
- Generates header only wrappers for EFL APIs

C++

- A core generator in the EFL
- Generates header only wrappers for EFL APIs
- Provides native C++-like object syntax

C++

SAMSUNG Open Source Group

- A core generator in the EFL
- Generates header only wrappers for EFL APIs
- Provides native C++-like object syntax
- Because of its header only nature, we don't care about ABI changes

C++

SAMSUNG Open Source Group

- A core generator in the EFL
- Generates header only wrappers for EFL APIs
- Provides native C++-like object syntax
- Because of its header only nature, we don't care about ABI changes
- Also doesn't change linkage over normal C API usage

C++

SAMSUNG Open Source Group

- A core generator in the EFL
- Generates header only wrappers for EFL APIs
- Provides native C++-like object syntax
- Because of its header only nature, we don't care about ABI changes
- Also doesn't change linkage over normal C API usage
- Interoperability with C API is also possible

Lua

Lua

- Also a core generator

Lua

- Also a core generator
- Written in Lua - easier string processing

Lua

- Also a core generator
- Written in Lua - easier string processing
- Requires no compiled code

Lua

- Also a core generator
- Written in Lua - easier string processing
- Requires no compiled code
- Loads EFL libraries at runtime

Lua

- Also a core generator
- Written in Lua - easier string processing
- Requires no compiled code
- Loads EFL libraries at runtime
- Requires a runtime

Lua

- Also a core generator
- Written in Lua - easier string processing
- Requires no compiled code
- Loads EFL libraries at runtime
- Requires a runtime
- Uses LuaJIT

Elua

Elua

- A library and a launcher for Lua EFL applications

Elua

- A library and a launcher for Lua EFL applications
- Provides some core functions needed by all applications

Elua

- A library and a launcher for Lua EFL applications
- Provides some core functions needed by all applications
- Small and lightweight

Elua

- A library and a launcher for Lua EFL applications
- Provides some core functions needed by all applications
- Small and lightweight
- Also offers various C utilities for state management

Elua

- A library and a launcher for Lua EFL applications
- Provides some core functions needed by all applications
- Small and lightweight
- Also offers various C utilities for state management
- Also offers i18n support

Eo objects and Lua

Eo objects and Lua

- We use LuaJIT FFI to access EFL API

Eo objects and Lua

- We use LuaJIT FFI to access EFL API
- FFI not exposed to apps - needs safe wrappers

Eo objects and Lua

- We use LuaJIT FFI to access EFL API
- FFI not exposed to apps - needs safe wrappers
- We don't want to generate too much boilerplate

Eo objects and Lua

- We use LuaJIT FFI to access EFL API
- FFI not exposed to apps - needs safe wrappers
- We don't want to generate too much boilerplate
- We generate simple declarative wrappers

Eo objects and Lua

- We use LuaJIT FFI to access EFL API
- FFI not exposed to apps - needs safe wrappers
- We don't want to generate too much boilerplate
- We generate simple declarative wrappers
- Calls are done using a special object runtime

The future

What's still not done?

Stabilize!

Stabilize!

- Stabilization is the primary goal

Stabilize!

- Stabilization is the primary goal
- Not happening for a few more releases

Stabilize!

- Stabilization is the primary goal
- Not happening for a few more releases
- Documentation is still ongoing task

Stabilize!

- Stabilization is the primary goal
- Not happening for a few more releases
- Documentation is still ongoing task
- We're unsure about handling ownership

Stabilize!

- Stabilization is the primary goal
- Not happening for a few more releases
- Documentation is still ongoing task
- We're unsure about handling ownership
- Functions and their binding still needs to be solved

Stabilize!

- Stabilization is the primary goal
- Not happening for a few more releases
- Documentation is still ongoing task
- We're unsure about handling ownership
- Functions and their binding still needs to be solved
- Refactor the implementation and fix all quirks

More testing and changes

More testing and changes

- We need more generators to help us test

More testing and changes

- We need more generators to help us test
- JavaScript V8 generator is coming up

More testing and changes

- We need more generators to help us test
- JavaScript V8 generator is coming up
- We also need to update all of EFL eo files

More testing and changes

- We need more generators to help us test
- JavaScript V8 generator is coming up
- We also need to update all of EFL eo files
- This should help uncover any potential problems

Thank you.

Daniel Kolesa
Samsung Open Source Group
d.kolesa@osg.samsung.com
@octaforge
FOSDEM 2016