lfe_comp(3)

Robert Virding

2008-2016

NAME

lfe_comp - Lisp Flavoured Erlang (LFE) compiler

SYNOPSIS

This module provides an interface to the standard LFE compiler. The compiler can handle files which contain multiple modules. It can generate either new files which contain the object code, or return binaries which can be loaded directly.

EXPORTS

file(FileName) -> CompRet

Is the same as `file(FileName, [report])`.

file(FileName, Options) -> CompRet

where

```
CompRet = ModRet | BinRet | ErrRet
ModRet = {ok,[ModOk]} | {ok,[ModOk],Warnings}
ModOk  = {ok,ModuleName} | {ok,ModuleName,Warnings}
BinRet = {ok,[ModBin]} | {ok,[ModBin],Warnings}
ModBin = {ok,ModuleName,Binary} | {ok,ModuleName,Binary,Warnings}
ErrRet = error | {error,[ModErr],Errors,Warnings}
ModErr = {error,Errors,Warnings}
```

Compile an LFE file, either writing the generated modules to files or returning them as binaries. The generated modules are ready to be loaded into Erlang.

The currently recognised options are:

- `binary` - Return the binary of the module and do not save it in a file.
- `no_docs`, `no-docs` - Do not parse docstrings and write the `"LDoc"` chunk in the binary of the module.
- `to_expand`, `to-expand` - Print a listing of the macro expanded LFE code in the file .expand. No object file is produced. Mainly useful for debugging and interest.
- `to_lint`, `to-lint` - Print a listing of the macro expanded and linted LFE code in the files .lint. No object files are produced. Mainly useful for debugging and interest.
- `to_erlang`, `to-erlang` - Print a listing of the Erlang AST in the file .erl. No object files are produced. Mainly useful for debugging and interest.
- `to_core0`, `to-core0`, `to_core`, `to-core` - Print a listing of the Core Erlang code before/after being optimised in the files .core. No object files are produced. Mainly useful for debugging and interest.
- `to_kernel`, `to-kernel` - Print a listing of the Kernel Erlang code in the files .kernel. No object files are produced. Mainly useful for debugging and interest.

- `to_asm`, `to-asm` - Print a listing of the Beam code in the files .S. No object files are produced. Mainly useful for debugging and interest.
- `{outdir,Dir}`, `[outdir,Dir]` - Save the generated files in director Dir instead of the current directory.
- `{i,Dir}`, `[i,Dir]` - Add dir to the list of directories to be searched when including a file.
- `report` - Print the errors and warnings as they occur.
- `return` - Return an extra return field containing Warnings on success or the errors and warnings in `{error,Errors,Warnings}` when there are errors.
- `debug_print`, `debug-print` - Causes the compiler to print a lot of debug information.
- `warnings_as_errors`, `warnings-as-errors` - Causes warnings to be treated as errors.
- `no_export_macros`, `no-export-macros` - Do not export macros from modules.

If the binary option is given then options that produce listing files will cause the internal formats for that compiler pass to be returned.

Both `Warnings` and `Errors` have the following format:

```
[{FileName,[ErrorInfo]}]
```

`ErrorInfo` is described below. When generating Errors and Warnings the line number is the line of the start of the form in which the error occurred. The file name has been included here to be compatible with the Erlang compiler. As yet there is no extra information about included files.

forms(Forms) -> CompRet

Is the same as forms(Forms, [report]).

forms(Forms, Options) -> CompRet

where

```
Forms = [sexpr()]
CompRet = BinRet | ErrRet
BinRet = {ok,[ModBin]} | {ok,[ModBin],Warnings}
ModBin = {ok,ModuleName,Binary} | {ok,ModuleName,Binary,Warnings}
ErrRet = error | {error,[ModErr],Errors,Warnings}
ModErr = {error,Errors,Warnings}
```

Compile the forms as an LFE module returning a binary. This function takes the same options as `lfe_comp:file/1/2`. When generating Errors and Warnings the "line number" is the index of the form in which the error occurred.

format_error(Error) -> Chars

Uses an ErrorDescriptor and returns a deep list of characters which describes the error. This function is usually called implicitly when an ErrorInfo structure is processed. See below.

ERROR INFORMATION

The `ErrorInfo` mentioned above is the standard `ErrorInfo` structure which is returned from all IO modules. It has the following format:

{ErrorLine,Module,ErrorDescriptor}

A string describing the error is obtained with the following call:

```
Module:format_error(ErrorDescriptor)
```

SEE ALSO

lfe_gen(3), **lfe_macro(3)**