

ferror

1.1.0

Generated by Doxygen 1.8.11

Contents

1	Main Page	1
1.1	Introduction	1
2	Modules Index	3
2.1	Modules List	3
3	Data Type Index	5
3.1	Data Types List	5
4	Module Documentation	7
4.1	error Module Reference	7
4.1.1	Detailed Description	8
4.1.2	Function/Subroutine Documentation	8
4.1.2.1	get_error_flag(this)	8
4.1.2.2	get_exit_on_error(this)	8
4.1.2.3	get_log_filename(this)	8
4.1.2.4	get_warning_flag(this)	9
4.1.2.5	has_error_occurred(this)	9
4.1.2.6	has_warning_occurred(this)	9
4.1.2.7	log_error(this, fcn, msg, flag)	9
4.1.2.8	report_error(this, fcn, msg, flag)	10
4.1.2.9	report_warning(this, fcn, msg, flag)	10
4.1.2.10	reset_error_status(this)	10
4.1.2.11	reset_warning_status(this)	10
4.1.2.12	set_exit_on_error(this, x)	11

4.1.2.13	set_log_filename(this, str)	11
4.2	ferror_c_binding Module Reference	11
4.2.1	Detailed Description	12
4.2.2	Function/Subroutine Documentation	12
4.2.2.1	alloc_errorhandler(obj)	12
4.2.2.2	cstr_2_fstr(cstr)	13
4.2.2.3	error_occurred(err)	13
4.2.2.4	fstr_2_cstr(fstr, cstr, csize)	13
4.2.2.5	get_error_code(err)	13
4.2.2.6	get_error_log_fname(err, fname, nfname)	14
4.2.2.7	get_errorhandler(obj, eobj)	14
4.2.2.8	get_exit_behavior(err)	14
4.2.2.9	get_warning_code(err)	14
4.2.2.10	register_error(err, fcn, msg, flag)	15
4.2.2.11	register_warning(err, fcn, msg, flag)	15
4.2.2.12	reset_error(err)	15
4.2.2.13	reset_warning(err)	15
4.2.2.14	set_error_log_fname(err, fname)	16
4.2.2.15	set_exit_behavior(err, x)	16
4.2.2.16	update_errorhandler(eobj, cobj)	16
4.2.2.17	warning_occurred(err)	16
4.2.2.18	write_error_log(err, fcn, msg, flag)	17
5	Data Type Documentation	19
5.1	ferror_c_binding::errorhandler Type Reference	19
5.1.1	Detailed Description	19
5.2	ferror::errors Type Reference	19
5.2.1	Detailed Description	20
	Index	21

Chapter 1

Main Page

1.1 Introduction

FERROR is a library to assist with error handling in Fortran projects.

Author

Jason Christopherson

Version

1.0

Chapter 2

Modules Index

2.1 Modules List

Here is a list of all documented modules with brief descriptions:

ferror	ferror	7
ferror_c_binding	ferror	11

Chapter 3

Data Type Index

3.1 Data Types List

Here are the data types with brief descriptions:

ferror_c_binding::errorhandler	A C compatible type encapsulating an errors object	19
ferror::errors	Defines a type for managing errors and warnings	19

Chapter 4

Module Documentation

4.1 ferror Module Reference

ferror

Data Types

- type [errors](#)
Defines a type for managing errors and warnings.

Functions/Subroutines

- pure character(len=:) function, allocatable [get_log_filename](#) (this)
Gets the name of the error log file.
- subroutine [set_log_filename](#) (this, str)
Sets the name of the error log file.
- subroutine [report_error](#) (this, fcn, msg, flag)
Reports an error condition to the user.
- subroutine [report_warning](#) (this, fcn, msg, flag)
Reports a warning message to the user.
- subroutine [log_error](#) (this, fcn, msg, flag)
Writes an error log file.
- pure logical function [has_error_occurred](#) (this)
Tests to see if an error has been encountered.
- subroutine [reset_error_status](#) (this)
Resets the error status flag to false, and the current error flag to zero.
- pure logical function [has_warning_occurred](#) (this)
Tests to see if a warning has been encountered.
- subroutine [reset_warning_status](#) (this)
Resets the warning status flag to false, and the current warning flag to zero.
- pure integer function [get_error_flag](#) (this)
Gets the current error flag.
- pure integer function [get_warning_flag](#) (this)
Gets the current warning flag.
- pure logical function [get_exit_on_error](#) (this)
Gets a logical value determining if the application should be terminated when an error is encountered.
- subroutine [set_exit_on_error](#) (this, x)
Sets a logical value determining if the application should be terminated when an error is encountered.

4.1.1 Detailed Description

ferro

Purpose

Provides a series of error codes and error handling mechanisms.

4.1.2 Function/Subroutine Documentation

4.1.2.1 pure integer function ferro::get_error_flag (class(errors), intent(in) *this*) [private]

Gets the current error flag.

Parameters

in	<i>this</i>	The errors object.
----	-------------	--------------------

Returns

The current error flag.

4.1.2.2 pure logical function ferro::get_exit_on_error (class(errors), intent(in) *this*) [private]

Gets a logical value determining if the application should be terminated when an error is encountered.

Parameters

in	<i>this</i>	The errors object.
----	-------------	--------------------

Returns

Returns true if the application should be terminated; else, false.

4.1.2.3 pure character(len = :) function, allocatable ferro::get_log_filename (class(errors), intent(in) *this*)

Gets the name of the error log file.

Parameters

in	<i>this</i>	The errors object.
----	-------------	--------------------

Returns

The filename.

4.1.2.4 pure integer function `ferror::get_warning_flag (class(errors), intent(in) this)` [private]

Gets the current warning flag.

Parameters

in	<i>this</i>	The errors object.
----	-------------	--------------------

Returns

The current warning flag.

4.1.2.5 pure logical function `ferror::has_error_occurred (class(errors), intent(in) this)` [private]

Tests to see if an error has been encountered.

Parameters

in	<i>this</i>	The errors object.
----	-------------	--------------------

Returns

Returns true if an error has been encountered; else, false.

4.1.2.6 pure logical function `ferror::has_warning_occurred (class(errors), intent(in) this)` [private]

Tests to see if a warning has been encountered.

Parameters

in	<i>this</i>	The errors object.
----	-------------	--------------------

Returns

Returns true if a warning has been encountered; else, false.

4.1.2.7 subroutine `ferror::log_error (class(errors), intent(in) this, character(len = *), intent(in) fcn, character(len = *), intent(in) msg, integer, intent(in) flag)` [private]

Writes an error log file.

Parameters

in	<i>this</i>	The errors object.
in	<i>fcn</i>	The name of the function or subroutine in which the error was encountered.
in	<i>msg</i>	The error message.
in	<i>flag</i>	The error flag.

4.1.2.8 subroutine `error::report_error` (`class(errors)`, `intent(inout) this`, `character(len = *)`, `intent(in) fcn`, `character(len = *)`, `intent(in) msg`, `integer`, `intent(in) flag`) `[private]`

Reports an error condition to the user.

Parameters

<code>in, out</code>	<code>this</code>	The errors object.
<code>in</code>	<code>fcn</code>	The name of the function or subroutine in which the error was encountered.
<code>in</code>	<code>msg</code>	The error message.
<code>in</code>	<code>flag</code>	The error flag.

Remarks

The default behavior prints an error message, appends the supplied information to a log file, and terminates the program.

4.1.2.9 subroutine `error::report_warning` (`class(errors)`, `intent(inout) this`, `character(len = *)`, `intent(in) fcn`, `character(len = *)`, `intent(in) msg`, `integer`, `intent(in) flag`) `[private]`

Reports a warning message to the user.

Parameters

<code>in, out</code>	<code>this</code>	The errors object.
<code>in</code>	<code>fcn</code>	The name of the function or subroutine from which the warning was issued.
<code>in</code>	<code>msg</code>	The warning message.
<code>in</code>	<code>flag</code>	The warning flag.

Remarks

The default behavior prints the warning message, and returns control back to the calling code.

4.1.2.10 subroutine `error::reset_error_status` (`class(errors)`, `intent(inout) this`) `[private]`

Resets the error status flag to false, and the current error flag to zero.

Parameters

<code>in, out</code>	<code>this</code>	The errors object.
----------------------	-------------------	--------------------

4.1.2.11 subroutine `error::reset_warning_status` (`class(errors)`, `intent(inout) this`) `[private]`

Resets the warning status flag to false, and the current warning flag to zero.

Parameters

<code>in, out</code>	<code>this</code>	The errors object.
----------------------	-------------------	--------------------

4.1.2.12 `subroutine error::set_exit_on_error (class(errors), intent(inout) this, logical, intent(in) x) [private]`

Sets a logical value determining if the application should be terminated when an error is encountered.

Parameters

<code>in, out</code>	<code>this</code>	The errors object.
<code>in</code>	<code>x</code>	Set to true if the application should be terminated when an error is reported; else, false.

4.1.2.13 `subroutine error::set_log_filename (class(errors), intent(inout) this, character(len = :), allocatable str) [private]`

Sets the name of the error log file.

Parameters

<code>in, out</code>	<code>this</code>	The errors object.
<code>in</code>	<code>str</code>	The filename.

4.2 `error_c_binding` Module Reference

`error`

Data Types

- type `errorhandler`
A C compatible type encapsulating an errors object.

Functions/Subroutines

- subroutine `alloc_errorhandler` (obj)
Initializes a new error handler object.
- subroutine `get_errorhandler` (obj, eobj)
Retrieves the errors object from the C compatible data structure.
- subroutine `update_errorhandler` (eobj, cobj)
Updates the errorhandler object.
- subroutine `get_error_log_fname` (err, fname, nfname)
Gets the name of the error log file.
- subroutine `set_error_log_fname` (err, fname)

- Sets the error log filename.*

 - subroutine `register_error` (err, fcn, msg, flag)

Reports an error condition to the user.

 - subroutine `register_warning` (err, fcn, msg, flag)

Reports a warning condition to the user.

 - subroutine `write_error_log` (err, fcn, msg, flag)

Writes an error log file.

 - logical(c_bool) function `error_occurred` (err)

Tests to see if an error has been encountered.

 - subroutine `reset_error` (err)

Resets the error status flag to false, and the current error flag to zero.

 - logical(c_bool) function `warning_occurred` (err)

Tests to see if a warning has been encountered.

 - subroutine `reset_warning` (err)

Resets the warning status flag to false, and the current warning flag to zero.

 - integer(c_int) function `get_error_code` (err)

Gets the current error flag.

 - integer(c_int) function `get_warning_code` (err)

Gets the current warning flag.

 - logical(c_bool) function `get_exit_behavior` (err)

Gets a logical value determining if the application should be terminated when an error is encountered.

 - subroutine `set_exit_behavior` (err, x)

Sets a logical value determining if the application should be terminated when an error is encountered.

 - character(len=:) function, allocatable `cstr_2_fstr` (cstr)

Copies a C string (null terminated) to a Fortran string.

 - subroutine `fstr_2_cstr` (fstr, cstr, csize)

Copies a Fortran string into a C string.

4.2.1 Detailed Description

ferror

Purpose

Provides C bindings to the ferror library.

4.2.2 Function/Subroutine Documentation

4.2.2.1 subroutine `ferror_c_binding::alloc_errorhandler` (type(errorhandler), intent(inout) *obj*)

Initializes a new error handler object.

Parameters

<i>in</i>	<i>obj</i>	The errorhandler object to allocate.
-----------	------------	--------------------------------------

4.2.2.2 `character(len = :)` function, allocatable `error_c_binding::cstr_2_fstr` (`character(kind = c_char)`, `dimension(*)`, `intent(in) cstr`)

Copies a C string (null terminated) to a Fortran string.

Parameters

<code>in</code>	<code>cstr</code>	The null-terminated C string.
-----------------	-------------------	-------------------------------

Returns

The Fortran copy.

4.2.2.3 `logical(c_bool)` function `error_c_binding::error_occurred` (`type(errorhandler)`, `intent(in) err`)

Tests to see if an error has been encountered.

Parameters

<code>in</code>	<code>err</code>	A pointer to the error handler object.
-----------------	------------------	--

Returns

Returns true if an error has been encountered; else, false.

4.2.2.4 subroutine `error_c_binding::fstr_2_cstr` (`character(len = *)`, `intent(in) fstr`, `character(kind = c_char)`, `dimension(*)`, `intent(out) cstr`, `integer`, `intent(inout) csize`)

Copies a Fortran string into a C string.

Parameters

<code>in</code>	<code>fstr</code>	The Fortran string to copy.
<code>out</code>	<code>cstr</code>	The null-terminated C string.
<code>in, out</code>	<code>csize</code>	On input, the size of the character buffer <code>cstr</code> . On output, the actual number of characters (not including the null character) written to <code>cstr</code> .

4.2.2.5 `integer(c_int)` function `error_c_binding::get_error_code` (`type(errorhandler)`, `intent(in) err`)

Gets the current error flag.

Parameters

<code>in</code>	<code>err</code>	The errorhandler object.
-----------------	------------------	--------------------------

Returns

The current error flag.

4.2.2.6 subroutine `ferror_c_binding::get_error_log_fname` (`type(errorhandler)`, `intent(in)` *err*, `character(kind = c_char)`, `dimension(*)`, `intent(out)` *fname*, `integer(c_int)`, `intent(inout)` *nfname*)

Gets the name of the error log file.

Parameters

<code>in</code>	<i>err</i>	The errorhandler object.
<code>out</code>	<i>fname</i>	A character buffer where the filename will be written. It is recommended that this be in the neighborhood of 256 elements.
<code>in, out</code>	<i>nfname</i>	On input, the actual size of the buffer. Be sure to leave room for the null terminator character. On output, the actual numbers of characters written to <i>fname</i> (not including the null character).

4.2.2.7 subroutine `ferror_c_binding::get_errorhandler` (`type(errorhandler)`, `intent(in)` *obj*, `class(errors)`, `intent(out)`, `allocatable` *eobj*)

Retrieves the errors object from the C compatible data structure.

Parameters

<code>in</code>	<i>obj</i>	The C compatible errorhandler data structure.
<code>out</code>	<i>eobj</i>	The resulting errors object.

4.2.2.8 `logical(c_bool)` function `ferror_c_binding::get_exit_behavior` (`type(errorhandler)`, `intent(in)` *err*)

Gets a logical value determining if the application should be terminated when an error is encountered.

Parameters

<code>in</code>	<i>err</i>	The errorhandler object.
-----------------	------------	--------------------------

Returns

Returns true if the application should be terminated; else, false.

4.2.2.9 `integer(c_int)` function `ferror_c_binding::get_warning_code` (`type(errorhandler)`, `intent(in)` *err*)

Gets the current warning flag.

Parameters

in	<i>err</i>	The errorhandler object.
----	------------	--------------------------

Returns

The current warning flag.

4.2.2.10 subroutine `ferror_c_binding::register_error` (`type(errorhandler)`, `intent(inout) err`, `character(kind = c_char)`, `intent(in) fcn`, `character(kind = c_char)`, `intent(in) msg`, `integer(c_int)`, `intent(in)`, `value flag`)

Reports an error condition to the user.

Parameters

in, out	<i>err</i>	A pointer to the error handler object.
in	<i>fcn</i>	The name of the function or subroutine in which the error was encountered.
in	<i>msg</i>	The error message.
in	<i>flag</i>	The error flag.

4.2.2.11 subroutine `ferror_c_binding::register_warning` (`type(errorhandler)`, `intent(inout) err`, `character(kind = c_char)`, `intent(in) fcn`, `character(kind = c_char)`, `intent(in) msg`, `integer(c_int)`, `intent(in)`, `value flag`)

Reports a warning condition to the user.

Parameters

in, out	<i>err</i>	The errorhandler object.
in	<i>fcn</i>	The name of the function or subroutine in which the warning was encountered.
in	<i>msg</i>	The warning message.
in	<i>flag</i>	The warning flag.

4.2.2.12 subroutine `ferror_c_binding::reset_error` (`type(errorhandler)`, `intent(inout) err`)

Resets the error status flag to false, and the current error flag to zero.

Parameters

in, out	<i>err</i>	The errorhandler object.
---------	------------	--------------------------

4.2.2.13 subroutine `ferror_c_binding::reset_warning` (`type(errorhandler)`, `intent(inout) err`)

Resets the warning status flag to false, and the current warning flag to zero.

Parameters

<i>in, out</i>	<i>err</i>	The errorhandler object.
----------------	------------	--------------------------

4.2.2.14 subroutine `ferror_c_binding::set_error_log_fname` (`type(errorhandler)`, `intent(inout) err`, `character(kind = c_char), dimension(*), intent(in) fname`)

Sets the error log filename.

Parameters

<i>in, out</i>	<i>err</i>	The errorhandler object.
<i>in</i>	<i>fname</i>	A null-terminated string containing the filename.

4.2.2.15 subroutine `ferror_c_binding::set_exit_behavior` (`type(errorhandler)`, `intent(inout) err`, `logical(c_bool), intent(in), value x`)

Sets a logical value determining if the application should be terminated when an error is encountered.

Parameters

<i>in, out</i>	<i>err</i>	The errorhandler object.
<i>in</i>	<i>x</i>	Set to true if the application should be terminated when an error is reported; else, false.

4.2.2.16 subroutine `ferror_c_binding::update_errorhandler` (`class(errors)`, `intent(in) eobj`, `type(errorhandler)`, `intent(inout) cobj`)

Updates the errorhandler object.

Parameters

<i>in</i>	<i>eobj</i>	The errors object.
<i>in, out</i>	<i>cobj</i>	The errorhandler object to update.

4.2.2.17 `logical(c_bool)` function `ferror_c_binding::warning_occurred` (`type(errorhandler)`, `intent(in) err`)

Tests to see if a warning has been encountered.

Parameters

<i>in</i>	<i>err</i>	The errorhandler object.
-----------	------------	--------------------------

Returns

Returns true if a warning has been encountered; else, false.

4.2.2.18 subroutine ferror_c_binding::write_error_log (type(errorhandler), intent(in) *err*, character(kind = c_char), intent(in) *fcn*, character(kind = c_char), intent(in) *msg*, integer(c_int), intent(in), value *flag*)

Writes an error log file.

Parameters

in	<i>err</i>	The errorhandler object.
in	<i>fcn</i>	The name of the function or subroutine in which the error was encountered.
in	<i>msg</i>	The error message.
in	<i>flag</i>	The error flag.

Chapter 5

Data Type Documentation

5.1 `error_c_binding::errorhandler` Type Reference

A C compatible type encapsulating an errors object.

Public Attributes

- `integer(c_int) n`
The size of the errors object, in bytes.
- `type(c_ptr) ptr`
A pointer to the errors object.

5.1.1 Detailed Description

A C compatible type encapsulating an errors object.

The documentation for this type was generated from the following file:

- `/home/jason/Documents/Code/error/src/error_c_binding.f90`

5.2 `error::errors` Type Reference

Defines a type for managing errors and warnings.

Public Member Functions

- procedure, public [get_log_filename](#)
Gets the name of the error log file.
- procedure, public [set_log_filename](#)
Sets the name of the error log file.
- procedure, public [report_error](#)
Reports an error condition to the user.
- procedure, public [report_warning](#)
Reports a warning message to the user.
- procedure, public [log_error](#)
Writes an error log file.
- procedure, public [has_error_occurred](#)
Tests to see if an error has been encountered.
- procedure, public [reset_error_status](#)
Resets the error status flag to false.
- procedure, public [has_warning_occurred](#)
Tests to see if a warning has been encountered.
- procedure, public [reset_warning_status](#)
Resets the warning status flag to false.
- procedure, public [get_error_flag](#)
Gets the current error flag.
- procedure, public [get_warning_flag](#)
Gets the current warning flag.
- procedure, public [get_exit_on_error](#)
Gets a logical value determining if the application should be terminated when an error is encountered.
- procedure, public [set_exit_on_error](#)
Sets a logical value determining if the application should be terminated when an error is encountered.

Public Attributes

- character(len=256) [m_fname](#) = "error_log.txt"
A maximum of 256 character error log filename.
- logical [m_founderror](#) = .false.
Found an error.
- logical [m_foundwarning](#) = .false.
Found a warning.
- integer [m_errorflag](#) = 0
The error flag.
- integer [m_warningflag](#) = 0
The warning flag.
- logical [m_exitonerror](#) = .true.
Terminate the application on error.

5.2.1 Detailed Description

Defines a type for managing errors and warnings.

The documentation for this type was generated from the following file:

- /home/jason/Documents/Code/ferror/src/ferror.f90

Index

- alloc_errorhandler
 - error_c_binding, 12
- cstr_2_fstr
 - error_c_binding, 12
- error_occurred
 - error_c_binding, 13
- ferror, 7
 - get_error_flag, 8
 - get_exit_on_error, 8
 - get_log_filename, 8
 - get_warning_flag, 8
 - has_error_occurred, 9
 - has_warning_occurred, 9
 - log_error, 9
 - report_error, 10
 - report_warning, 10
 - reset_error_status, 10
 - reset_warning_status, 10
 - set_exit_on_error, 11
 - set_log_filename, 11
- ferror::errors, 19
- error_c_binding, 11
 - alloc_errorhandler, 12
 - cstr_2_fstr, 12
 - error_occurred, 13
 - fstr_2_cstr, 13
 - get_error_code, 13
 - get_error_log_fname, 14
 - get_errorhandler, 14
 - get_exit_behavior, 14
 - get_warning_code, 14
 - register_error, 15
 - register_warning, 15
 - reset_error, 15
 - reset_warning, 15
 - set_error_log_fname, 16
 - set_exit_behavior, 16
 - update_errorhandler, 16
 - warning_occurred, 16
 - write_error_log, 17
- error_c_binding::errorhandler, 19
- fstr_2_cstr
 - error_c_binding, 13
- get_error_code
 - error_c_binding, 13
- get_error_flag
 - error, 8
- get_error_log_fname
 - error_c_binding, 14
- get_errorhandler
 - error_c_binding, 14
- get_exit_behavior
 - error_c_binding, 14
- get_exit_on_error
 - error, 8
- get_log_filename
 - error, 8
- get_warning_code
 - error_c_binding, 14
- get_warning_flag
 - error, 8
- has_error_occurred
 - error, 9
- has_warning_occurred
 - error, 9
- log_error
 - error, 9
- register_error
 - error_c_binding, 15
- register_warning
 - error_c_binding, 15
- report_error
 - error, 10
- report_warning
 - error, 10
- reset_error
 - error_c_binding, 15
- reset_error_status
 - error, 10
- reset_warning
 - error_c_binding, 15
- reset_warning_status
 - error, 10
- set_error_log_fname
 - error_c_binding, 16
- set_exit_behavior
 - error_c_binding, 16
- set_exit_on_error
 - error, 11
- set_log_filename
 - error, 11
- update_errorhandler

`error_c_binding`, [16](#)

`warning_occurred`

`error_c_binding`, [16](#)

`write_error_log`

`error_c_binding`, [17](#)