

ferror

1.2.1

Generated by Doxygen 1.8.11

Contents

1	Main Page	1
1.1	Introduction	1
2	Modules Index	2
2.1	Modules List	2
3	Data Type Index	2
3.1	Data Types List	2
4	Module Documentation	2
4.1	ferror Module Reference	2
4.1.1	Detailed Description	3
4.1.2	Function/Subroutine Documentation	3
4.2	ferror_c_binding Module Reference	7
4.2.1	Detailed Description	8
4.2.2	Function/Subroutine Documentation	8
5	Data Type Documentation	13
5.1	ferror_c_binding::errorhandler Type Reference	13
5.1.1	Detailed Description	13
5.2	ferror::errors Type Reference	14
5.2.1	Detailed Description	15
	Index	17

1 Main Page

1.1 Introduction

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

Author

Jason Christopherson

Version

1.2.1

2 Modules Index

2.1 Modules List

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

ferror	
ferror	2
ferror_c_binding	
ferror	7

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	13
ferror::errors	
Defines a type for managing errors and warnings	14

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.
- pure logical function `get_suppress_printing` (this)
Gets a logical value determining if printing of error and warning messages should be suppressed.
- subroutine `set_suppress_printing` (this, x)
Sets a logical value determining if printing of error and warning messages should be suppressed.

4.1.1 Detailed Description

error

Purpose

Provides a series of error codes and error handling mechanisms.

4.1.2 Function/Subroutine Documentation

4.1.2.1 pure integer function `error::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 `error::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 `error::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 logical function `error::get_suppress_printing (class(errors), intent(in) this)` [private]

Gets a logical value determining if printing of error and warning messages should be suppressed.

Parameters

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

Returns

True if message printing should be suppressed; else, false to allow printing.

4.1.2.5 pure integer function `error::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.

4.1.2.6 pure logical function `fehler::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.

4.1.2.7 pure logical function `fehler::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.

4.1.2.8 subroutine `fehler::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.

4.1.2.9 subroutine `fehler::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

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

Parameters

in	<i>msg</i>	The error message.
in	<i>flag</i>	The error flag.

Remarks

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

4.1.2.10 `subroutine ferror::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

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

Remarks

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

4.1.2.11 `subroutine ferror::reset_error_status (class(errors), intent(inout) this) [private]`

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

Parameters

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

4.1.2.12 `subroutine ferror::reset_warning_status (class(errors), intent(inout) this) [private]`

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

Parameters

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

4.1.2.13 `subroutine ferror::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.

4.1.2.14 `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.

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

Sets a logical value determining if printing of error and warning messages should be suppressed.

Parameters

<code>in, out</code>	<code>this</code>	The errors object.
<code>in</code>	<code>x</code>	Set to true if message printing should be suppressed; else, false to allow printing.

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_log_filename` (`err`, `fname`, `nfname`)
Gets the name of the error log file.
- subroutine `set_log_filename` (`err`, `fname`)
Sets the error log filename.
- subroutine `report_error` (`err`, `fcn`, `msg`, `flag`)

- Reports an error condition to the user.*
- subroutine [report_warning](#) (err, fcn, msg, flag)
- Reports a warning condition to the user.*
- subroutine [log_error](#) (err, fcn, msg, flag)
- Writes an error log file.*
- logical(c_bool) function [has_error_occurred](#) (err)
- Tests to see if an error has been encountered.*
- subroutine [reset_error_status](#) (err)
- Resets the error status flag to false, and the current error flag to zero.*
- logical(c_bool) function [has_warning_occurred](#) (err)
- Tests to see if a warning has been encountered.*
- subroutine [reset_warning_status](#) (err)
- Resets the warning status flag to false, and the current warning flag to zero.*
- integer(c_int) function [get_error_flag](#) (err)
- Gets the current error flag.*
- integer(c_int) function [get_warning_flag](#) (err)
- Gets the current warning flag.*
- logical(c_bool) function [get_exit_on_error](#) (err)
- Gets a logical value determining if the application should be terminated when an error is encountered.*
- subroutine [set_exit_on_error](#) (err, x)
- Sets a logical value determining if the application should be terminated when an error is encountered.*
- logical(c_bool) function [get_suppress_printing](#) (err)
- Gets a logical value determining if printing of error and warning messages should be suppressed.*
- subroutine [set_suppress_printing](#) (err, x)
- Sets a logical value determining if printing of error and warning messages should be suppressed.*
- 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

in	<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 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>cszize</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> .

4.2.2.4 `integer(c_int)` function `error_c_binding::get_error_flag` (`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.

4.2.2.5 subroutine `error_c_binding::get_errorhandler` (`type(errorhandler)`, `intent(in)`, `target obj`, `class(errors)`, `intent(out)`, allocatable `eobj`)

Retrieves the errors object from the C compatible data structure.

Parameters

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

4.2.2.6 `logical(c_bool)` function `error_c_binding::get_exit_on_error (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>	<code>err</code>	The errorhandler object.
-----------------	------------------	--------------------------

Returns

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

4.2.2.7 subroutine `error_c_binding::get_log_filename (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>	<code>err</code>	The errorhandler object.
<code>out</code>	<code>fname</code>	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>	<code>nfname</code>	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 <code>fname</code> (not including the null character).

4.2.2.8 `logical(c_bool)` function `error_c_binding::get_suppress_printing (type(errorhandler), intent(in) err)`

Gets a logical value determining if printing of error and warning messages should be suppressed.

Parameters

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

Returns

True if message printing should be suppressed; else, false to allow printing.

4.2.2.9 `integer(c_int)` function `error_c_binding::get_warning_flag (type(errorhandler), intent(in) err)`

Gets the current warning flag.

Parameters

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

Returns

The current warning flag.

4.2.2.10 `logical(c_bool) function error_c_binding::has_error_occurred (type(errorhandler), intent(in) err)`

Tests to see if an error has been encountered.

Parameters

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

Returns

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

4.2.2.11 `logical(c_bool) function error_c_binding::has_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.

4.2.2.12 `subroutine error_c_binding::log_error (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

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

4.2.2.13 `subroutine error_c_binding::report_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

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

Parameters

in	<i>msg</i>	The error message.
in	<i>flag</i>	The error flag.

4.2.2.14 `subroutine ferror_c_binding::report_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.

4.2.2.15 `subroutine ferror_c_binding::reset_error_status (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.
---------	------------	--------------------------

4.2.2.16 `subroutine ferror_c_binding::reset_warning_status (type(errorhandler), intent(inout) err)`

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

Parameters

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

4.2.2.17 `subroutine ferror_c_binding::set_exit_on_error (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

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

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

Sets the error log filename.

Parameters

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

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

Sets a logical value determining if printing of error and warning messages should be suppressed.

Parameters

in, out	<i>err</i>	The errorhandler object.
in	<i>x</i>	Set to true if message printing should be suppressed; else, false to allow printing.

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

Updates the errorhandler object.

Parameters

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

5 Data Type Documentation

5.1 `ferror_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/ferror/src/ferror_c_binding.f90`

5.2 `ferror::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.
- procedure, public `get_suppress_printing`
Gets a logical value determining if printing of error and warning messages should be suppressed.
- procedure, public `set_suppress_printing`
Sets a logical value determining if printing of error and warning messages should be suppressed.

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.
- logical `m_suppressprinting` = .false.
Suppress printing of error and warning messages.

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, 8
- cstr_2_fstr
 - error_c_binding, 8
- error, 2
 - get_error_flag, 3
 - get_exit_on_error, 4
 - get_log_filename, 4
 - get_suppress_printing, 4
 - get_warning_flag, 4
 - has_error_occurred, 5
 - has_warning_occurred, 5
 - log_error, 5
 - report_error, 5
 - report_warning, 6
 - reset_error_status, 6
 - reset_warning_status, 6
 - set_exit_on_error, 6
 - set_log_filename, 7
 - set_suppress_printing, 7
- error::errors, 14
- error_c_binding, 7
 - alloc_errorhandler, 8
 - cstr_2_fstr, 8
 - fstr_2_cstr, 9
 - get_error_flag, 9
 - get_errorhandler, 9
 - get_exit_on_error, 9
 - get_log_filename, 10
 - get_suppress_printing, 10
 - get_warning_flag, 10
 - has_error_occurred, 11
 - has_warning_occurred, 11
 - log_error, 11
 - report_error, 11
 - report_warning, 12
 - reset_error_status, 12
 - reset_warning_status, 12
 - set_exit_on_error, 12
 - set_log_filename, 12
 - set_suppress_printing, 13
 - update_errorhandler, 13
- error_c_binding::errorhandler, 13
- fstr_2_cstr
 - error_c_binding, 9
- get_error_flag
 - error, 3
 - error_c_binding, 9
- get_errorhandler
 - error_c_binding, 9
- get_exit_on_error
 - error, 4
 - error_c_binding, 9
- get_log_filename
 - error, 4
 - error_c_binding, 10
- get_suppress_printing
 - error, 4
 - error_c_binding, 10
- get_warning_flag
 - error, 4
 - error_c_binding, 10
- has_error_occurred
 - error, 5
 - error_c_binding, 11
- has_warning_occurred
 - error, 5
 - error_c_binding, 11
- log_error
 - error, 5
 - error_c_binding, 11
- report_error
 - error, 5
 - error_c_binding, 11
- report_warning
 - error, 6
 - error_c_binding, 12
- reset_error_status
 - error, 6
 - error_c_binding, 12
- reset_warning_status
 - error, 6
 - error_c_binding, 12
- set_exit_on_error
 - error, 6
 - error_c_binding, 12
- set_log_filename
 - error, 7
 - error_c_binding, 12
- set_suppress_printing
 - error, 7
 - error_c_binding, 13
- update_errorhandler
 - error_c_binding, 13