ferror

1.3.1

Generated by Doxygen 1.8.11

Contents

1 Main Page

1.1 Introduction

FERROR is a library to assist with error handling in Fortran projects. The error handling capabilities also have been extended to be called from C thereby providing both an error handling mechanism for C projects as well as allowing C interop with Fortran projects that use this library to handle errors.

Example

The following piece of code offers a simple introduction to the use of this library.

```
program example
    use ferror
    use, intrinsic :: iso_fortran_env, only : int32
    implicit none

    ! Variables
    type(errors) :: err_mgr

    ! Ensure the error reporting doesn't terminate the application.  The default
    ! behavior terminates the application.
    call err_mgr%set_exit_on_error(.false.)

    ! Don't print the error message to the command line.  The default behavior
    ! prints the error information to the command line.
    call err_mgr%set_suppress_printing(.true.)

    ! Call the routine that causes the error
    call causes_error(err_mgr)

    ! Print the error information
    print '(A)', "An error occurred in the following subroutine: " // &
        err_mgr%get_error_fcn_name()
    print '(A)', "The error message is: " // err_mgr%get_error_message()
    print '(AI0)', "The error code is: ", err_mgr%get_error_flag()
contains

! The purpose of this subroutine is to simply trigger an error condition.
subroutine causes_error(err)
    ! Arguments
    class(errors), intent(inout) :: err

    ! Define an error flag
    integer(int32), parameter :: error_flag = 200

    ! Trigger the error condition
    call err%report_error(&
        "causes_error", &                   ! The subroutine or function name
        "This is a test error message.", &  ! The error message.
        error_flag)                         ! The error flag
end subroutine

end program
```

The above program produces the following output.

```
An error occurred in the following subroutine: causes_error
The error message is: This is a test error message.
The error code is: 200
```

The above program also creates a log file. The log file is titled error_log.txt by default, but can be named whatever by the user. The contents of the file written from the above program are as follows.

```
***** ERROR *****
1/2/2018; 16:49:40
Function: causes_error
Error Flag: 200
Message:
This is a test error message.
```

If additional errors are encountered, the information is simply appended to the end of the file.

The same example above can be written in C. The C implementation is as follows.

```c
#include <stdio.h>
#include "ferror.h"

void causes_error(errorhandler *err);

int main(void) {
    // Variables
    errorhandler err_mgr;
    char fname[256], msg[256];
    int flag, fnamelength = 256, msglength = 256;

    // Initialization
    alloc_errorhandler(&err_mgr);

    // Ensure the error reporting doesn't terminate the application
    set_exit_on_error(&err_mgr, false);

    // Don't print the error message to the command line
    set_suppress_printing(&err_mgr, true);

    // Call the routine that causes the error
    causes_error(&err_mgr);

    // Retrieve the error information
    get_error_fcn_name(&err_mgr, fname, &fnamelength);
    get_error_message(&err_mgr, msg, &msglength);
    flag = get_error_flag(&err_mgr);

    // Print the error information
    printf("An error occurred in the following subroutine: %s\nThe error message is: %s\nThe error code is:
       %i\n",
        fname, msg, flag);

    // End
    free_errorhandler(&err_mgr);
    return 0;
}

void causes_error(errorhandler *err) {
    report_error(err,                       // The errorhandler object
        "causes_error",                     // The function name
        "This is a test error message.",    // The error message
        200);                               // The error flag
}
```

The above C program produces exactly the same output as the Fortran example.

2 Modules Index

2.1 Modules List

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

3 Data Type Index

3.1 Data Types List

Here are the data types with brief descriptions:

4 Module Documentation

4.1 ferror Module Reference

ferror

Data Types

- interface error_callback

  Defines the signature of routine that can be used to clean up after an error condition is encountered.
- type errors

  Defines a type for managing errors and warnings.

Functions/Subroutines

- pure character(len=:) function, allocatable er_get_log_filename (this)

  Gets the name of the error log file.
- subroutine er_set_log_filename (this, str)

  Sets the name of the error log file.
- subroutine er_report_error (this, fcn, msg, flag, obj)

  Reports an error condition to the user.
- subroutine er_report_warning (this, fcn, msg, flag)

  Reports a warning message to the user.
- subroutine er_log_error (this, fcn, msg, flag)

  Writes an error log file.
- pure logical function er_has_error_occurred (this)

  Tests to see if an error has been encountered.
- subroutine er_reset_error_status (this)

  Resets the error status flag to false, and the current error flag to zero.
- pure logical function er_has_warning_occurred (this)

  Tests to see if a warning has been encountered.
- subroutine er_reset_warning_status (this)

  Resets the warning status flag to false, and the current warning flag to zero.
- pure integer(int32) function er_get_error_flag (this)

  Gets the current error flag.
- pure integer(int32) function er_get_warning_flag (this)

  Gets the current warning flag.
- pure logical function er_get_exit_on_error (this)

  Gets a logical value determining if the application should be terminated when an error is encountered.
- subroutine er_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 er_get_suppress_printing (this)

  Gets a logical value determining if printing of error and warning messages should be suppressed.
- subroutine er_set_suppress_printing (this, x)

  Sets a logical value determining if printing of error and warning messages should be suppressed.
- character(len=:) function, allocatable er_get_err_msg (this)

  Gets the current error message.
- character(len=:) function, allocatable er_get_warning_msg (this)

  Gets the current warning message.
- character(len=:) function, allocatable er_get_err_fcn (this)

  Gets the name of the routine that initiated the error.
- character(len=:) function, allocatable er_get_warning_fcn (this)

  Gets the name of the routine that initiated the warning.
- procedure(error_callback) function, pointer er_get_err_fcn_ptr (this)

  Gets the routine to call when an error has been logged.
- subroutine er_set_err_fcn_ptr (this, ptr)

  Sets the routine to call when an error has been logged.

4.1.1 Detailed Description

ferror

Purpose

Provides a series of error codes and error handling mechanisms.

4.1.2 Function/Subroutine Documentation

4.1.2.1 character(len = :) function, allocatable ferror::er_get_err_fcn (class(errors), intent(in) *this*) [private]

Gets the name of the routine that initiated the error.

Parameters

in	*this*	The errors object.

Returns

The routine or function name.

4.1.2.2 procedure(error_callback) function, pointer ferror::er_get_err_fcn_ptr (class(errors), intent(in) *this*) [private]

Gets the routine to call when an error has been logged.

Parameters

in	*this*	The errors object.

Returns

A pointer to the routine.

4.1.2.3 character(len = :) function, allocatable ferror::er_get_err_msg (class(errors), intent(in) *this*) [private]

Gets the current error message.

Parameters

in	*this*	The errors object.

Returns

The error message.

4.1.2.4 pure integer(int32) function ferror::er_get_error_flag (class(errors), intent(in) *this*) [private]

Gets the current error flag.

Parameters

in	*this*	The errors object.

Returns

The current error flag.

4.1.2.5 pure logical function ferror::er_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	*this*	The errors object.

Returns

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

4.1.2.6 pure character(len = :) function, allocatable ferror::er_get_log_filename (class(errors), intent(in) *this*) [private]

Gets the name of the error log file.

Parameters

in	*this*	The errors object.

Returns

The filename.

4.1.2.7 pure logical function ferror::er_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	*this*	The errors object.

Returns

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

4.1.2.8 character(len = :) function, allocatable ferror::er_get_warning_fcn (class(errors), intent(in) *this*) [private]

Gets the name of the routine that initiated the warning.

Parameters

in	*this*	The errors object.

Returns

The routine or function name.

4.1.2.9 pure integer(int32) function ferror::er_get_warning_flag (class(errors), intent(in) *this*) [private]

Gets the current warning flag.

Parameters

in	*this*	The errors object.

Returns

The current warning flag.

4.1.2.10 character(len = :) function, allocatable ferror::er_get_warning_msg (class(errors), intent(in) *this*) [private]

Gets the current warning message.

Parameters

in	*this*	The errors object.

Returns

The warning message.

4.1.2.11 pure logical function ferror::er_has_error_occurred (class(errors), intent(in) *this*) [private]

Tests to see if an error has been encountered.

Parameters

in	*this*	The errors object.

Returns

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

4.1.2.12 pure logical function ferror::er_has_warning_occurred (class(errors), intent(in) *this*) [private]

Tests to see if a warning has been encountered.

Parameters

in	*this*	The errors object.

Returns

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

4.1.2.13 subroutine ferror::er_log_error (class(errors), intent(in) *this,* character(len = ∗), intent(in) *fcn,* character(len = ∗), intent(in) *msg,* integer(int32), intent(in) *flag*) [private]

Writes an error log file.

Parameters

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

4.1.2.14 subroutine ferror::er_report_error (class(errors), intent(inout) *this,* character(len = ∗), intent(in) *fcn,* character(len = ∗), intent(in) *msg,* integer(int32), intent(in) *flag,* class(∗), intent(inout), optional *obj*) [private]

Reports an error condition to the user.

Parameters

in,out	*this*	The errors object.
in	*fcn*	The name of the function or subroutine in which the error was encountered.

Parameters

in	*msg*	The error message.
in	*flag*	The error flag.
in,out	*obj*	An optional unlimited polymorphic object that can be passed to provide information to the clean-up routine.

Remarks

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

4.1.2.15 subroutine ferror::er_report_warning (class(errors), intent(inout) *this,* character(len = ∗), intent(in) *fcn,* character(len = ∗), intent(in) *msg,* integer(int32), intent(in) *flag*) [private]

Reports a warning message to the user.

Parameters

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

Remarks

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

4.1.2.16 subroutine ferror::er_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	*this*	The errors object.

4.1.2.17 subroutine ferror::er_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	*this*	The errors object.

4.1.2.18 subroutine ferror::er_set_err_fcn_ptr (class(errors), intent(inout) *this,* procedure(error_callback), intent(in), pointer *ptr*) [private]

Sets the routine to call when an error has been logged.

Parameters

in,out	*this*	The errors object.
in	*ptr*	A pointer to the routine.

4.1.2.19 subroutine ferror::er_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

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

4.1.2.20 subroutine ferror::er_set_log_filename (class(errors), intent(inout) *this,* character(len = ∗), intent(in) *str*) [private]

Sets the name of the error log file.

Parameters

in,out	*this*	The errors object.
in	*str*	The filename.

4.1.2.21 subroutine ferror::er_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

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

4.2 ferror_c_binding Module Reference

ferror

Data Types

- interface c_error_callback
- type callback_manager

  A type that allows C interop with the Fortran error callback routine.
- type errorhandler

  A C compatible type encapsulating an errors object.

Functions/Subroutines

- subroutine, public alloc_errorhandler (obj)

  Initializes a new error handler object.
- subroutine, public free_errorhandler (obj)

  Frees resources held by the errorhandler object.
- subroutine, public get_errorhandler (obj, eobj)

  Retrieves the errors object from the C compatible data structure.
- subroutine, public get_log_filename (err, fname, nfname)

  Gets the name of the error log file.
- subroutine, public set_log_filename (err, fname)

  Sets the error log filename.
- subroutine, public report_error (err, fcn, msg, flag)

  Reports an error condition to the user.
- subroutine, public report_warning (err, fcn, msg, flag)

  Reports a warning condition to the user.
- subroutine, public log_error (err, fcn, msg, flag)

  Writes an error log file.
- logical(c_bool) function, public has_error_occurred (err)

  Tests to see if an error has been encountered.
- subroutine, public reset_error_status (err)

  Resets the error status flag to false, and the current error flag to zero.
- logical(c_bool) function, public has_warning_occurred (err)

  Tests to see if a warning has been encountered.
- subroutine, public reset_warning_status (err)

  Resets the warning status flag to false, and the current warning flag to zero.
- integer(c_int) function, public get_error_flag (err)

  Gets the current error flag.
- integer(c_int) function, public get_warning_flag (err)

  Gets the current warning flag.
- logical(c_bool) function, public get_exit_on_error (err)

  Gets a logical value determining if the application should be terminated when an error is encountered.
- subroutine, public 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, public get_suppress_printing (err)

  Gets a logical value determining if printing of error and warning messages should be suppressed.
- subroutine, public set_suppress_printing (err, x)

  Sets a logical value determining if printing of error and warning messages should be suppressed.
- subroutine, public get_error_message (err, msg, nmsg)

Gets the current error message.

- subroutine, public get_warning_message (err, msg, nmsg)

  Gets the current warning message.
- subroutine, public get_error_fcn_name (err, fname, nfname)

  Gets the name of the function or subroutine that issued the last error message.
- subroutine, public get_warning_fcn_name (err, fname, nfname)

  Gets the name of the function or subroutine that issued the last warning message.
- subroutine, public report_error_with_callback (err, fcn, msg, flag, cback, args)

  Reports an error condition to the user, and executes a callback routine.
- 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.
- subroutine **err_callback** (this, obj)

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, public ferror_c_binding::alloc_errorhandler (type(errorhandler), intent(inout) *obj*)

Initializes a new error handler object.

Parameters

in	*obj*	The errorhandler object to allocate.

4.2.2.2 character(len = :) function, allocatable ferror_c_binding::cstr_2_fstr (character(kind = c_char), dimension(*), intent(in) *cstr*) [private]

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

Parameters

in	*cstr*	The null-terminated C string.

Returns

The Fortran copy.

4.2.2.3 subroutine, public ferror_c_binding::free_errorhandler (type(errorhandler), intent(inout), target *obj*)

Frees resources held by the errorhandler object.

Parameters

in,out	*obj*	The errorhandler object.

4.2.2.4 subroutine ferror_c_binding::fstr_2_cstr (character(len = ∗), intent(in) *fstr,* character(kind = c_char), dimension(∗), intent(out) *cstr,* integer, intent(inout) *csize*) [private]

Copies a Fortran string into a C string.

Parameters

in	*fstr*	The Fortran string to copy.
out	*cstr*	The null-terminated C string.
in,out	*csize*	On input, the size of the character buffer `cstr`. On output, the actual number of characters (not including the null character) writen to `cstr`.

4.2.2.5 subroutine, public ferror_c_binding::get_error_fcn_name (type(errorhandler), intent(in) *err,* character(kind = c_char), dimension(∗), intent(out) *fname,* integer(c_int), intent(inout) *nfname*)

Gets the name of the function or subroutine that issued the last error message.

Parameters

in	*err*	The errorhandler object.
out	*fname*	A character buffer where the name will be written.
in,out	*nfname*	On input, the actual size of the buffer. On output, the actual number of characters written to `fname` (not including the null character).

4.2.2.6 integer(c_int) function, public ferror_c_binding::get_error_flag (type(errorhandler), intent(in) *err*)

Gets the current error flag.

Parameters

in	*err*	The errorhandler object.

Returns

The current error flag.

4.2.2.7 subroutine, public ferror_c_binding::get_error_message (type(errorhandler), intent(in) *err,* character(kind = c_char), dimension(*), intent(out) *msg,* integer(c_int), intent(inout) *nmsg*)

Gets the current error message.

Parameters

in	*err*	The errorhandler object.
out	*mst*	A character buffer where the message will be written.
in,out	*nmsg*	On input, the actual size of the buffer. On output, the actual number of characters written to msg (not including the null character).

4.2.2.8 subroutine, public ferror_c_binding::get_errorhandler (type(errorhandler), intent(in), target *obj,* type(errors), intent(out), pointer *eobj*)

Retrieves the errors object from the C compatible data structure.

Parameters

in	*obj*	The C compatible errorhandler data structure.
out	*eobj*	The resulting errors object.

4.2.2.9 logical(c_bool) function, public ferror_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

in	*err*	The errorhandler object.

Returns

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

4.2.2.10 subroutine, public ferror_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

in	*err*	The errorhandler object.

Parameters

out	*fname*	A character buffer where the filename will be written. It is recommended that this be in the neighborhood of 256 elements.
in,out	*nfname*	On input, the actual size of the buffer. Be sure to leave room for the null terminator character. On output, the actual number of characters written to `fname` (not including the null character).

4.2.2.11 logical(c_bool) function, public ferror_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

in	*err*	The errorhandler object.

Returns

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

4.2.2.12 subroutine, public ferror_c_binding::get_warning_fcn_name (type(errorhandler), intent(in) *err,* character(kind = c_char), dimension(*), intent(out) *fname,* integer(c_int), intent(inout) *nfname*)

Gets the name of the function or subroutine that issued the last warning message.

Parameters

in	*err*	The errorhandler object.
out	*fname*	A character buffer where the name will be written.
in,out	*nfname*	On input, the actual size of the buffer. On output, the actual number of characters written to `fname` (not including the null character).

4.2.2.13 integer(c_int) function, public ferror_c_binding::get_warning_flag (type(errorhandler), intent(in) *err*)

Gets the current warning flag.

Parameters

in	*err*	The errorhandler object.

Returns

The current warning flag.

4.2.2.14 subroutine, public ferror_c_binding::get_warning_message (type(errorhandler), intent(in) *err,* character(kind = c_char), dimension(*), intent(out) *msg,* integer(c_int), intent(inout) *nmsg*)

Gets the current warning message.

Parameters

in	*err*	The errorhandler object.
out	*mst*	A character buffer where the message will be written.
in,out	*nmsg*	On input, the actual size of the buffer. On output, the actual number of characters written to msg (not including the null character).

4.2.2.15 logical(c_bool) function, public ferror_c_binding::has_error_occurred (type(errorhandler), intent(in) *err*)

Tests to see if an error has been encountered.

Parameters

in	*err*	A pointer to the error handler object.

Returns

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

4.2.2.16 logical(c_bool) function, public ferror_c_binding::has_warning_occurred (type(errorhandler), intent(in) *err*)

Tests to see if a warning has been encountered.

Parameters

in	*err*	The errorhandler object.

Returns

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

4.2.2.17 subroutine, public ferror_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

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

4.2.2.18 subroutine, public ferror_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

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

4.2.2.19 subroutine, public ferror_c_binding::report_error_with_callback (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,* type(c_funptr), intent(in), value *cback,* type(c_ptr), intent(in), value *args*)

Reports an error condition to the user, and executes a callback routine.

Parameters

in,out	*err*	A pointer to the error handler object.
in	*fcn*	The name of the function or subroutine in which the error was encountered.
in	*msg*	The error message.
in	*flag*	The error flag.
in	*cback*	A pointer to the callback function.
in	*args*	A pointer to an object to pass to the callback function.

4.2.2.20 subroutine, public 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	*err*	The errorhandler object.
in	*fcn*	The name of the function or subroutine in which the warning was encountered.
in	*msg*	The warning message.
in	*flag*	The warning flag.

4.2.2.21 subroutine, public 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	*err*	The errorhandler object.

4.2.2.22 subroutine, public 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	*err*	The errorhandler object.

4.2.2.23 subroutine, public 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	*err*	The errorhandler object.
in	*x*	Set to true if the application should be terminated when an error is reported; else, false.

4.2.2.24 subroutine, public 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	*err*	The errorhandler object.
in	*fname*	A null-terminated string containing the filename.

4.2.2.25 subroutine, public 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	*err*	The errorhandler object.
in	*x*	Set to true if message printing should be suppressed; else, false to allow printing.

5 Data Type Documentation

5.1 ferror_c_binding::c_error_callback Interface Reference

Private Member Functions

- subroutine **c_error_callback** (ptr)

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

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

5.2 ferror_c_binding::callback_manager Type Reference

A type that allows C interop with the Fortran error callback routine.

Private Attributes

- procedure(c_error_callback), pointer, nopass fcn
  A pointer to the C callback routine.
- type(c_ptr) args
  A pointer to the C-supplied arguments.

5.2.1 Detailed Description

A type that allows C interop with the Fortran error callback routine.

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

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

5.3 ferror::error_callback Interface Reference

Defines the signature of routine that can be used to clean up after an error condition is encountered.

Private Member Functions

- subroutine **error_callback** (err, obj)

5.3.1 Detailed Description

Defines the signature of routine that can be used to clean up after an error condition is encountered.

Parameters

`in`	*err*	The errors-based object managing the error handling tasks.
`in,out`	*obj*	An unlimited polymorphic object that can be passed to provide information to the clean-up routine.

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

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

5.4 ferror_c_binding::errorhandler Type Reference

A C compatible type encapsulating an errors object.

Private Attributes

- type(c_ptr) ptr

  A pointer to the errors object.
- integer(c_int) n

  The size of the errors object, in bytes.

5.4.1 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

5.5 ferror::errors Type Reference

Defines a type for managing errors and warnings.

Public Member Functions

- procedure, public get_log_filename => er_get_log_filename

  Gets the name of the error log file.
- procedure, public set_log_filename => er_set_log_filename

  Sets the name of the error log file.
- procedure, public report_error => er_report_error

  eports an error condition to the user.
- procedure, public report_warning => er_report_warning

  Reports a warning message to the user.
- procedure, public log_error => er_log_error

  Writes an error log file.
- procedure, public has_error_occurred => er_has_error_occurred

  Tests to see if an error has been encountered.
- procedure, public reset_error_status => er_reset_error_status

  Resets the error status flag to false.
- procedure, public has_warning_occurred => er_has_warning_occurred

  Tests to see if a warning has been encountered.
- procedure, public reset_warning_status => er_reset_warning_status

  Resets the warning status flag to false.
- procedure, public get_error_flag => er_get_error_flag

  Gets the current error flag.
- procedure, public get_warning_flag => er_get_warning_flag

  Gets the current warning flag.
- procedure, public get_exit_on_error => er_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 => er_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 => er_get_suppress_printing

  Gets a logical value determining if printing of error and warning messages should be suppressed.
- procedure, public set_suppress_printing => er_set_suppress_printing

  Sets a logical value determining if printing of error and warning messages should be suppressed.
- procedure, public get_error_message => er_get_err_msg

  Gets the currently error message.
- procedure, public get_warning_message => er_get_warning_msg

  Gets the current warning message.
- procedure, public get_error_fcn_name => er_get_err_fcn

  Gets the name of the routine that initiated the error.
- procedure, public get_warning_fcn_name => er_get_warning_fcn

  Gets the name of the routine that initiated the warning.
- procedure, public get_clean_up_routine => er_get_err_fcn_ptr

  Gets the routine to call when an error has been logged.
- procedure, public set_clean_up_routine => er_set_err_fcn_ptr

  Sets the routine to call when an error has been logged.

Private 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(int32) m_errorflag = 0

  The error flag.
- integer(int32) 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.
- character(len=:), allocatable m_errormessage

  The error message.
- character(len=:), allocatable m_warningmessage

  The warning message.
- character(len=:), allocatable m_efunname

  The function where the error occurred.
- character(len=:), allocatable m_wfunname

  The function where the warning occurred.
- procedure(error_callback), pointer, pass m_errcleanup => null()

  A pointer to a routine that can be called upon notice of an error.

5.5.1 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