

**ferror**

1.2.0

Generated by Doxygen 1.8.11



# Contents

<b>1</b>	<b>Main Page</b>	<b>1</b>
1.1	Introduction . . . . .	1
<b>2</b>	<b>Modules Index</b>	<b>3</b>
2.1	Modules List . . . . .	3
<b>3</b>	<b>Data Type Index</b>	<b>5</b>
3.1	Data Types List . . . . .	5
<b>4</b>	<b>Module Documentation</b>	<b>7</b>
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_suppress_printing(this) . . . . .	9
4.1.2.5	get_warning_flag(this) . . . . .	9
4.1.2.6	has_error_occurred(this) . . . . .	9
4.1.2.7	has_warning_occurred(this) . . . . .	10
4.1.2.8	log_error(this, fcn, msg, flag) . . . . .	10
4.1.2.9	report_error(this, fcn, msg, flag) . . . . .	10
4.1.2.10	report_warning(this, fcn, msg, flag) . . . . .	10
4.1.2.11	reset_error_status(this) . . . . .	11
4.1.2.12	reset_warning_status(this) . . . . .	11

4.1.2.13	set_exit_on_error(this, x)	11
4.1.2.14	set_log_filename(this, str)	11
4.1.2.15	set_suppress_printing(this, x)	12
4.2	ferror_c_binding Module Reference	12
4.2.1	Detailed Description	13
4.2.2	Function/Subroutine Documentation	13
4.2.2.1	alloc_errorhandler(obj)	13
4.2.2.2	cstr_2_fstr(cstr)	13
4.2.2.3	fstr_2_cstr(fstr, cstr, csize)	14
4.2.2.4	get_error_flag(err)	14
4.2.2.5	get_errorhandler(obj, eobj)	14
4.2.2.6	get_exit_on_error(err)	14
4.2.2.7	get_log_filename(err, fname, nfname)	15
4.2.2.8	get_suppress_printing(err)	15
4.2.2.9	get_warning_flag(err)	15
4.2.2.10	has_error_occurred(err)	15
4.2.2.11	has_warning_occurred(err)	16
4.2.2.12	log_error(err, fcn, msg, flag)	16
4.2.2.13	report_error(err, fcn, msg, flag)	16
4.2.2.14	report_warning(err, fcn, msg, flag)	16
4.2.2.15	reset_error_status(err)	17
4.2.2.16	reset_warning_status(err)	17
4.2.2.17	set_exit_on_error(err, x)	17
4.2.2.18	set_log_filename(err, fname)	17
4.2.2.19	set_suppress_printing(err, x)	18
4.2.2.20	update_errorhandler(eobj, cobj)	18
<b>5</b>	<b>Data Type Documentation</b>	<b>19</b>
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	21
	<b>Index</b>	<b>23</b>

# Chapter 1

## Main Page

### 1.1 Introduction

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

**Author**

Jason Christopherson

**Version**

1.2.0



## Chapter 2

# Modules Index

### 2.1 Modules List

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

<a href="#">ferror</a>	<b>ferror</b> . . . . .	<a href="#">7</a>
<a href="#">ferror_c_binding</a>	<b>ferror</b> . . . . .	<a href="#">12</a>





## Chapter 3

# Data Type Index

### 3.1 Data Types List

Here are the data types with brief descriptions:

<a href="#">ferror_c_binding::errorhandler</a>	A C compatible type encapsulating an errors object . . . . .	19
<a href="#">ferror::errors</a>	Defines a type for managing errors and warnings . . . . .	19



## Chapter 4

# Module Documentation

### 4.1 `error` Module Reference

#### `error`

##### 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	this	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	this	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**

<code>in</code>	<code>this</code>	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**

<code>in</code>	<code>this</code>	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**

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

**Returns**

The current warning flag.

**4.1.2.6 pure logical function `error::has_error_occurred ( class(errors), intent(in) this )` [private]**

Tests to see if an error has been encountered.

**Parameters**

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

**Returns**

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

#### 4.1.2.7 pure logical function `error::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 `error::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 `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

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.
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 `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

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

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

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

Sets the name of the error log file.

## Parameters

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

#### **error**

#### Purpose

Provides C bindings to the `error` library.

### 4.2.2 Function/Subroutine Documentation

#### 4.2.2.1 subroutine `error_c_binding::alloc_errorhandler` ( `type(errorhandler)`, `intent(inout) obj` )

Initializes a new error handler object.

#### Parameters

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

**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>	<i>err</i>	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>	<i>obj</i>	The C compatible errorhandler data structure.
<code>out</code>	<i>eobj</i>	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>	<i>err</i>	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

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

**Returns**

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

4.2.2.11 `logical(c_bool) function ferror_c_binding::has_warning_occurred ( type(errorhandler), intent(in) err )`

Tests to see if a warning has been encountered.

**Parameters**

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

**Returns**

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

4.2.2.12 `subroutine 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	<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.

4.2.2.13 `subroutine 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	<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.

4.2.2.14 `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

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

4.2.2.15 `subroutine error_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

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

4.2.2.16 `subroutine error_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

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

4.2.2.17 `subroutine error_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

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

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

Sets the error log filename.

## Parameters

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

**4.2.2.19** subroutine `error_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**

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

**4.2.2.20** subroutine `error_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.

## 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.*
- 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, 13
- cstr\_2\_fstr
  - error\_c\_binding, 13
- ferror, 7
  - get\_error\_flag, 8
  - get\_exit\_on\_error, 8
  - get\_log\_filename, 8
  - get\_suppress\_printing, 9
  - get\_warning\_flag, 9
  - has\_error\_occurred, 9
  - has\_warning\_occurred, 9
  - log\_error, 10
  - report\_error, 10
  - report\_warning, 10
  - reset\_error\_status, 11
  - reset\_warning\_status, 11
  - set\_exit\_on\_error, 11
  - set\_log\_filename, 11
  - set\_suppress\_printing, 12
- ferror::errors, 19
- error\_c\_binding, 12
  - alloc\_errorhandler, 13
  - cstr\_2\_fstr, 13
  - fstr\_2\_cstr, 13
  - get\_error\_flag, 14
  - get\_errorhandler, 14
  - get\_exit\_on\_error, 14
  - get\_log\_filename, 14
  - get\_suppress\_printing, 15
  - get\_warning\_flag, 15
  - has\_error\_occurred, 15
  - has\_warning\_occurred, 16
  - log\_error, 16
  - report\_error, 16
  - report\_warning, 16
  - reset\_error\_status, 17
  - reset\_warning\_status, 17
  - set\_exit\_on\_error, 17
  - set\_log\_filename, 17
  - set\_suppress\_printing, 17
  - update\_errorhandler, 18
- error\_c\_binding::errorhandler, 19
- fstr\_2\_cstr
  - error\_c\_binding, 13
- get\_error\_flag
  - error, 8
- error\_c\_binding, 14
  - get\_errorhandler
    - error\_c\_binding, 14
  - get\_exit\_on\_error
    - error, 8
    - error\_c\_binding, 14
  - get\_log\_filename
    - error, 8
    - error\_c\_binding, 14
  - get\_suppress\_printing
    - error, 9
    - error\_c\_binding, 15
  - get\_warning\_flag
    - error, 9
    - error\_c\_binding, 15
  - has\_error\_occurred
    - error, 9
    - error\_c\_binding, 15
  - has\_warning\_occurred
    - error, 9
    - error\_c\_binding, 16
  - log\_error
    - error, 10
    - error\_c\_binding, 16
  - report\_error
    - error, 10
    - error\_c\_binding, 16
  - report\_warning
    - error, 10
    - error\_c\_binding, 16
  - reset\_error\_status
    - error, 11
    - error\_c\_binding, 17
  - reset\_warning\_status
    - error, 11
    - error\_c\_binding, 17
  - set\_exit\_on\_error
    - error, 11
    - error\_c\_binding, 17
  - set\_log\_filename
    - error, 11
    - error\_c\_binding, 17
  - set\_suppress\_printing
    - error, 12
    - error\_c\_binding, 17
  - update\_errorhandler

`ferror_c_binding`, [18](#)