Reference Manual

Generated by Doxygen 1.7.4

Wed Mar 14 2012 18:57:00

Contents

Chapter 1

Example of Object-Oriented C

Author

Matthew J. Miller (matt@matthewjmiller.net)

Date

2011

1.1 Introduction

This gives an idea of how aspects of object-oriented programming can be implmented in C. It particular, it demonstrates multiple inheritance, an abstract class, and multiple levels of inheritance. Opaque pointers are used to completely encapsulate a class's virtual function table so it is inaccessible directly even to friend classes.

It and of itself, this isn't particularly useful because there is so much manual work to setup basic OO-relations. This is more of a playground to try stuff for which a code generator could then be built to automate the code for these relationships.

`base1` and `base2` are base classes. `base2` is an abstract class that has a pure virtual function (`base2_increase_val1()`). `derived1` inherits from both `base1` and `base2`. `derived2` inherits from `derived1`.

The files follow the convention:

- **base1.h:** Public API for class `base1`.
- **base1_friend.h:** Friend API for class `base1`. Should only be included by classes that inherit from `base1`.
- **base1.c:** Implementation of `base1` and includes private data not directly accessible by even friend classes.

1.2 Usage

Just run `make` and the `test_c_oo` test program will be run. You can edit `test_c_oo.c` to try various things with this class hierarchy. Running `make clean` will remove the executable and .o files.

1.3 GNU General Public License

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

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

Chapter 4

Class Documentation

4.1 base1_private_st_ Struct Reference

Public Attributes

- const base1_vtable_st * vtable

4.1.1 Detailed Description

Private variables which cannot be directly accessed by any other class including children.

Definition at line 33 of file base1.c.

4.1.2 Member Data Documentation

4.1.2.1 const base1_vtable_st* base1_private_st_::vtable

Virtual function table

Definition at line 35 of file base1.c.

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

- base1.c

4.2 base1_public_data_st_ Struct Reference

```
#include <base1.h>
```

Public Attributes

- uint8_t val1
- uint32_t val2

4.2.1 Detailed Description

Public data for the class

Definition at line 33 of file base1.h.

4.2.2 Member Data Documentation

4.2.2.1 uint8_t base1_public_data_st_::val1

Some value

Definition at line 35 of file base1.h.

4.2.2.2 uint32_t base1_public_data_st_::val2

Some other value

Definition at line 37 of file base1.h.

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

- base1.h

4.3 base1_st_ Struct Reference

```
#include <base1_friend.h>
```

Public Attributes

- base1_private_handle private_h
- base1_public_data_st public_data
- uint32_t val3

4.3.1 Detailed Description

Friend accessible data for this class

Definition at line 34 of file base1_friend.h.

4.3.2 Member Data Documentation

4.3.2.1 base1_private_handle base1_st_::private_h

Reference to private data

Definition at line 36 of file base1_friend.h.

4.3.2.2 base1_public_data_st base1_st_::public_data

Public data

Definition at line 38 of file base1_friend.h.

4.3.2.3 uint32_t base1_st_::val3

Some value

Definition at line 40 of file base1_friend.h.

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

- base1_friend.h

4.4 base1_vtable_st_ Struct Reference

```
#include <base1_friend.h>
```

Public Attributes

- base1_delete_fn delete_fn
- base1_type_string_fn type_string_fn
- base1_string_fn string_fn
- base1_string_size_fn string_size_fn
- base1_increase_val3_fn increase_val3_fn

4.4.1 Detailed Description

The virtual table to be specified by friend classes.

See also

base1_set_vtable()

Definition at line 78 of file base1_friend.h.

4.4.2 Member Data Documentation

4.4.2.1 base1_delete_fn base1_vtable_st_::delete_fn

Function to delete object

Definition at line 80 of file base1_friend.h.

4.4.2.2 base1_increase_val3_fn base1_vtable_st_::increase_val3_fn

Function to increase val3

Definition at line 88 of file base1_friend.h.

4.4.2.3 base1_string_fn base1_vtable_st_::string_fn

Function to give string for object state

Definition at line 84 of file base1_friend.h.

4.4.2.4 base1_string_size_fn base1_vtable_st_::string_size_fn

Function to give size to use for object state string

Definition at line 86 of file base1_friend.h.

4.4.2.5 base1_type_string_fn base1_vtable_st_::type_string_fn

Function to give string for object type

Definition at line 82 of file base1_friend.h.

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

- base1_friend.h

4.5 base2_private_st_ Struct Reference

Public Attributes

- const base2_vtable_st * vtable

4.5.1 Detailed Description

Private variables which cannot be directly accessed by any other class including children.

Definition at line 35 of file base2.c.

4.5.2 Member Data Documentation

4.5.2.1 const base2_vtable_st* base2_private_st_::vtable

Virtual function table

Definition at line 37 of file base2.c.

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

- base2.c

4.6 base2_st_ Struct Reference

```
#include <base2_friend.h>
```

Public Attributes

- base2_private_handle private_h
- uint32_t val1

4.6.1 Detailed Description

Friend accessible data for this class

Definition at line 34 of file base2_friend.h.

4.6.2 Member Data Documentation

4.6.2.1 base2_private_handle base2_st_::private_h

Reference to private data

Definition at line 36 of file base2_friend.h.

4.6.2.2 uint32_t base2_st_::val1

Some value

Definition at line 38 of file base2_friend.h.

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

- base2_friend.h

4.7 base2_vtable_st_ Struct Reference

`#include <base2_friend.h>`

Public Attributes

- base2_delete_fn delete_fn
- base2_type_string_fn type_string_fn
- base2_string_fn string_fn
- base2_string_size_fn string_size_fn
- base2_increase_val1_fn increase_val1_fn

4.7.1 Detailed Description

The virtual table to be specified by friend classes.

See also

base2_set_vtable()

Definition at line 76 of file base2_friend.h.

4.7.2 Member Data Documentation

4.7.2.1 base2_delete_fn base2_vtable_st_::delete_fn

Function to delete object

Definition at line 78 of file base2_friend.h.

4.7.2.2 base2_increase_val1_fn base2_vtable_st_::increase_val1_fn

Function to increase val1

Definition at line 86 of file base2_friend.h.

4.7.2.3 base2_string_fn base2_vtable_st_::string_fn

Function to give string for object state

Definition at line 82 of file base2_friend.h.

4.7.2.4 base2_string_size_fn base2_vtable_st_::string_size_fn

Function to give size to use for object state string

Definition at line 84 of file base2_friend.h.

4.7.2.5 base2_type_string_fn base2_vtable_st_::type_string_fn

Function to give string for object type

Definition at line 80 of file base2_friend.h.

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

- base2_friend.h

4.8 derived1_private_st_ Struct Reference

Public Attributes

- const derived1_vtable_st * vtable

4.8.1 Detailed Description

Private variables which cannot be directly accessed by any other class including children.

Definition at line 33 of file derived1.c.

4.8.2 Member Data Documentation

4.8.2.1 const derived1_vtable_st* derived1_private_st_::vtable

Virtual function table

Definition at line 35 of file derived1.c.

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

- derived1.c

4.9 derived1_st_ Struct Reference

```
#include <derived1_friend.h>
```

Public Attributes

- derived1_private_handle private_h
- base1_st base1
- base2_st base2
- uint32_t val4

4.9.1 Detailed Description

Friend accessible data for this class

Definition at line 36 of file derived1_friend.h.

4.9.2 Member Data Documentation

4.9.2.1 base1_st derived1_st_::base1

Inherited base1 state

Definition at line 40 of file derived1_friend.h.

4.9.2.2 base2_st derived1_st_::base2

Inherited base2 state

Definition at line 42 of file derived1_friend.h.

4.9.2.3 derived1_private_handle derived1_st_::private_h

Reference to private data

Definition at line 38 of file derived1_friend.h.

4.9.2.4 uint32_t derived1_st_::val4

Some value

Definition at line 44 of file derived1_friend.h.

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

- derived1_friend.h

4.10 derived1_vtable_st_ Struct Reference

`#include <derived1_friend.h>`

Public Attributes

- base1_vtable_st * base1_vtable
- base2_vtable_st * base2_vtable
- derived1_delete_fn delete_fn
- derived1_increase_val4_fn increase_val4_fn

4.10.1 Detailed Description

The virtual table to be specified by friend classes.

See also

derived1_set_vtable()

Definition at line 64 of file derived1_friend.h.

4.10.2 Member Data Documentation

4.10.2.1 base1_vtable_st* derived1_vtable_st_::base1_vtable

Pointer to the base1 functions to use

Definition at line 66 of file derived1_friend.h.

4.10.2.2 base2_vtable_st* derived1_vtable_st_::base2_vtable

Pointer to the base2 functions to use

Definition at line 68 of file derived1_friend.h.

4.10.2.3 derived1_delete_fn derived1_vtable_st_::delete_fn

Function to delete object

Definition at line 70 of file derived1_friend.h.

4.10.2.4 derived1_increase_val4_fn derived1_vtable_st_::increase_val4_fn

Function to increase val4

Definition at line 72 of file derived1_friend.h.

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

- derived1_friend.h

4.11 derived2_st_ Struct Reference

Public Attributes

- derived1_st derived1

4.11.1 Detailed Description

Private data for this class

Definition at line 31 of file derived2.c.

4.11.2 Member Data Documentation

4.11.2.1 derived1_st derived2_st_::derived1

Inherited derived1 state

Definition at line 33 of file derived2.c.

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

- derived2.c

Chapter 5

File Documentation

5.1 base1.c File Reference

```
#include "base1_friend.h"
```

Classes

- struct base1_private_st_

Defines

- #define BASE1_STR_SIZE 128

Typedefs

- typedef struct base1_private_st_ base1_private_st

Functions

- const char * base1_get_val1_description (void)
- my_rc_e base1_get_public_data (base1_handle base1_h, base1_public_data_st *public_data)
- my_rc_e base1_set_public_data (base1_handle base1_h, base1_public_data_st *public_data)
- my_rc_e base1_string_size (base1_handle base1_h, size_t *buffer_size)
- const char * base1_type_string (base1_handle base1_h)
- my_rc_e base1_string (base1_handle base1_h, char *buffer, size_t buffer_size)
- void base1_friend_delete (base1_handle base1_h)
- void base1_delete (base1_handle base1_h)
- my_rc_e base1_increase_val3 (base1_handle base1_h)

- my_rc_e base1_inherit_vtable (const base1_vtable_st *parent_vtable, base1_vtable_st *child_vtable, bool do_null_check)
- my_rc_e base1_set_vtable (base1_handle base1_h, base1_vtable_st *vtable)
- my_rc_e base1_init (base1_handle base1_h)
- base1_handle base1_new1 (void)
- base1_handle base1_new2 (base1_public_data_st *public_data)
- base1_handle base1_new3 (uint8_t val1, uint32_t val3)

5.1.1 Detailed Description

Author

Matt Miller <matt@matthewjmiller.net>

5.1.2 LICENSE

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

5.1.3 DESCRIPTION

This is the implements a base class from which children class may inherit.

Definition in file base1.c.

5.1.4 Define Documentation

5.1.4.1 #define BASE1_STR_SIZE 128

Size for this object to use for base1_string_size_fn

Definition at line 27 of file base1.c.

5.1.5 Typedef Documentation

5.1.5.1 typedef struct base1_private_st_ base1_private_st

Private variables which cannot be directly accessed by any other class including children.

5.1.6 Function Documentation

5.1.6.1 void base1_delete (base1_handle *base1_h*)

Delete the object. This is a virtual function. Upon return, the object is not longer valid.

Parameters

base1_h	The object. If NULL, then this function is a no-op.

Definition at line 307 of file base1.c.

5.1.6.2 void base1_friend_delete (base1_handle *base1_h*)

Allow a friend class to delete the base1 object. It is assumed that the friend class is managing the memory for the base1 object and, thus, the object will not be freed. However, members within the base1 object may be freed. This does not call the virtual function table version of delete, but rather the delete specifically for type base1.

Parameters

base1_h	The object. If NULL, then this function is a no-op.

See also

base1_delete()

Definition at line 281 of file base1.c.

5.1.6.3 my_rc_e base1_get_public_data (base1_handle *base1_h,* base1_public_data_st * *public_data*)

Gets a copy of the public data for the given object. Note this is a shallow copy of the data, modifying it will not change the object's state. Writing the object's state is handled seperately by base1_set_public_data().

Parameters

base1_h	The object
public_data	The data buffer into which the values should be read

Returns

Return code

See also

base1_set_public_data()

Definition at line 59 of file base1.c.

5.1.6.4 const char* base1_get_val1_description (void)

Example of a static class method. It takes no instance of an object.

Returns

Description of val1

Definition at line 43 of file base1.c.

5.1.6.5 my_rc_e base1_increase_val3 (base1_handle *base1_h*)

Increase val3 for the object. This is a virtual function.

Parameters

base1_h	The object

Returns

Return code

Definition at line 347 of file base1.c.

5.1.6.6 my_rc_e base1_inherit_vtable (const base1_vtable_st * *parent_vtable,* base1_vtable_st * *child_vtable,* bool *do_null_check*)

Fill in the child vtable with values inherited from the parent_vtable for all functions left NULL in the child vtable.

Parameters

parent_vtable	The parent vtable from which to inherit.
child_vtable	The child vtable to which functions may be inherited.
do_null_check	Indicates whether an error should be thrown if a function in the child vtable is NULL after inheritance.

Definition at line 382 of file base1.c.

5.1.6.7 my_rc_e base1_init (base1_handle *base1_h*)

Allows a friend class to initialize their inner base1 object. Must be called before the base1 object is used. If an error is returned, any clean-up was handled internally and there is no need to call a delete function.

Parameters

base1_h	The object

Returns

Return code

See also

base1_delete()
base1_friend_delete()

Definition at line 461 of file base1.c.

5.1.6.8 base1_handle base1_new1 (void)

Create a new base1 object.

Returns

The object or NULL if creation failed

Definition at line 499 of file base1.c.

5.1.6.9 base1_handle base1_new2 (base1_public_data_st * *public_data*)

Create a new base1 object.

Parameters

public_data	The initial public data for the new object.

Returns

The object or NULL if creation failed

Definition at line 529 of file base1.c.

5.1.6.10 base1_handle base1_new3 (uint8_t *val1*, uint32_t *val3*)

Create a new base1 object.

Parameters

val1	The initial val1 for the new object.
val3	The initial val3 for the new object.

Returns

The object or NULL if creation failed

Definition at line 554 of file base1.c.

5.1.6.11 my_rc_e base1_set_public_data (base1_handle *base1_h*, base1_public_data_st * *public_data*)

Set the public data for the given object. Note that this creates a deep copy of the data in the object. Also note that it overwrites all public data in the object, not certain fields seletively.

Parameters

base1_h	The object
public_data	The data buffer whose values should be written into the object

Returns

Return code

See also

base1_get_public_data()

Definition at line 85 of file base1.c.

5.1.6.12 my_rc_e base1_set_vtable (base1_handle *base1_h*, base1_vtable_st * *vtable*)

This is a function used by friend classes to set the virtual table according to which methods they wish to override.

Parameters

base1_h	The object
vtable	The virtual table specification for the friend class. If a function pointer is NULL, then the base1 function is inherited.

Returns

Return code

Definition at line 425 of file base1.c.

5.1.6.13 my_rc_e base1_string (base1_handle *base1_h*, char * *buffer*, size_t *buffer_size*)

Get a string representation of the object. This is a virtual function.

Parameters

base1_h	The object
buffer	The buffer in which to put the string.
buffer_size	The size of the buffer.

Returns

Return code

See also

base1_string_size()

Definition at line 186 of file base1.c.

5.1.6.14 my_rc_e base1_string_size (base1_handle *base1_h*, size_t * *buffer_size*)

Get the minimum size of a string buffer that should be used to get a string representation of the object. This is a virtual function.

Parameters

base1_h	The object
buffer_size	Outputs the size of the buffer that should be used.

Returns

Return code

See also

base1_string()

Definition at line 109 of file base1.c.

5.1.6.15 const char* base1_type_string (base1_handle *base1_h*)

Get the string describing the type of the object. This is a virtual function.

Parameters

base1_h	The object

Returns

The string indicating the object type.

Definition at line 164 of file base1.c.

5.2 base1.h File Reference

```
#include "common.h"
```

Classes

- struct base1_public_data_st_

Typedefs

- typedef struct base1_st_ * base1_handle
- typedef struct base1_public_data_st_ base1_public_data_st

Functions

- const char * base1_get_val1_description (void)
- my_rc_e base1_get_public_data (base1_handle base1_h, base1_public_data_st *public_data)
- my_rc_e base1_set_public_data (base1_handle base1_h, base1_public_data_st *public_data)
- my_rc_e base1_increase_val3 (base1_handle base1_h)
- base1_handle base1_new1 (void)
- base1_handle base1_new2 (base1_public_data_st *public_data)
- base1_handle base1_new3 (uint8_t val1, uint32_t val3)
- void base1_delete (base1_handle base1_h)
- const char * base1_type_string (base1_handle base1_h)
- my_rc_e base1_string (base1_handle base1_h, char *buffer, size_t buffer_size)
- my_rc_e base1_string_size (base1_handle base1_h, size_t *buffer_size)

5.2.1 Detailed Description

Author

Matt Miller <matt@matthewjmiller.net>

5.2.2 LICENSE

5.2.3 DESCRIPTION

This is the public interface for base1 class.

Definition in file base1.h.

5.2.4 Typedef Documentation

5.2.4.1 typedef struct base1_st_* base1_handle

Opaque pointer to reference instances of this class

Definition at line 30 of file base1.h.

5.2.4.2 typedef struct base1_public_data_st_ base1_public_data_st

Public data for the class

5.2.5 Function Documentation

5.2.5.1 void base1_delete (base1_handle *base1_h*)

Delete the object. This is a virtual function. Upon return, the object is not longer valid.

Parameters

base1_h	The object. If NULL, then this function is a no-op.

Definition at line 307 of file base1.c.

5.2.5.2 my_rc_e base1_get_public_data (base1_handle *base1_h*, base1_public_data_st * *public_data*)

Gets a copy of the public data for the given object. Note this is a shallow copy of the data, modifying it will not change the object's state. Writing the object's state is handled seperately by base1_set_public_data().

Parameters

base1_h	The object
public_data	The data buffer into which the values should be read

Returns

Return code

See also

base1_set_public_data()

Definition at line 59 of file base1.c.

5.2.5.3 const char* base1_get_val1_description (void)

Example of a static class method. It takes no instance of an object.

Returns

Description of val1

Definition at line 43 of file base1.c.

5.2.5.4 my_rc_e base1_increase_val3 (base1_handle *base1_h*)

Increase val3 for the object. This is a virtual function.

Parameters

base1_h	The object

Returns

Return code

Definition at line 347 of file base1.c.

5.2.5.5 base1_handle base1_new1 (void)

Create a new base1 object.

Returns

The object or NULL if creation failed

Definition at line 499 of file base1.c.

5.2.5.6 base1_handle base1_new2 (base1_public_data_st * *public_data*)

Create a new base1 object.

Parameters

public_data	The initial public data for the new object.

Returns

The object or NULL if creation failed

Definition at line 529 of file base1.c.

5.2.5.7 base1_handle base1_new3 (uint8_t *val1*, uint32_t *val3*)

Create a new base1 object.

Parameters

val1	The initial val1 for the new object.
val3	The initial val3 for the new object.

Returns

The object or NULL if creation failed

Definition at line 554 of file base1.c.

5.2.5.8 my_rc_e base1_set_public_data (base1_handle *base1_h*, base1_public_data_st * *public_data*)

Set the public data for the given object. Note that this creates a deep copy of the data in the object. Also note that it overwrites all public data in the object, not certain fields seletively.

Parameters

base1_h	The object
public_data	The data buffer whose values should be written into the object

Returns

Return code

See also

base1_get_public_data()

Definition at line 85 of file base1.c.

5.2.5.9 my_rc_e base1_string (base1_handle *base1_h*, char * *buffer*, size_t *buffer_size*)

Get a string representation of the object. This is a virtual function.

Parameters

base1_h	The object
buffer	The buffer in which to put the string.
buffer_size	The size of the buffer.

Returns

Return code

See also

base1_string_size()

Definition at line 186 of file base1.c.

5.2.5.10 my_rc_e base1_string_size (base1_handle *base1_h*, size_t * *buffer_size*)

Get the minimum size of a string buffer that should be used to get a string representation of the object. This is a virtual function.

Parameters

base1_h	The object
buffer_size	Outputs the size of the buffer that should be used.

Returns

Return code

See also

base1_string()

Definition at line 109 of file base1.c.

5.2.5.11 const char* base1_type_string (base1_handle *base1_h*)

Get the string describing the type of the object. This is a virtual function.

Parameters

base1_h	The object

Returns

The string indicating the object type.

Definition at line 164 of file base1.c.

5.3 base1_friend.h File Reference

```
#include "base1.h"
```

Classes

- struct base1_st_
- struct base1_vtable_st_

Typedefs

- typedef struct base1_private_st_ * base1_private_handle
- typedef struct base1_st_ base1_st
- typedef void(* base1_delete_fn)(base1_handle base1_h)
- typedef const char *(* base1_type_string_fn)(base1_handle base1_h)
- typedef my_rc_e(* base1_string_fn)(base1_handle base1_h, char *buffer, size_t buffer_size)
- typedef my_rc_e(* base1_string_size_fn)(base1_handle base1_h, size_t *buffer_size)
- typedef my_rc_e(* base1_increase_val3_fn)(base1_handle base1_h)
- typedef struct base1_vtable_st_ base1_vtable_st

Functions

- my_rc_e base1_inherit_vtable (const base1_vtable_st *parent_vtable, base1_vtable_st *child_vtable, bool do_null_check)
- my_rc_e base1_set_vtable (base1_handle base1_h, base1_vtable_st *vtable)
- void base1_friend_delete (base1_handle base1_h)
- my_rc_e base1_init (base1_handle base1_h)

5.3.1 Detailed Description

Author

Matt Miller <matt@matthewjmiller.net>

5.3.2 LICENSE

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

5.3.3 DESCRIPTION

This is the friend interface for base1 class. It should only be included by sub-classes of base1.

Definition in file base1_friend.h.

5.3.4 Typedef Documentation

5.3.4.1 typedef void(∗ base1_delete_fn)(base1_handle base1_h)

Virtual function declaration.

Definition at line 47 of file base1_friend.h.

5.3.4.2 typedef my_rc_e(∗ base1_increase_val3_fn)(base1_handle base1_h)

Virtual function declaration.

Definition at line 71 of file base1_friend.h.

5.3.4.3 typedef struct base1_private_st_∗ base1_private_handle

Opaque pointer to reference private data for the class

Definition at line 31 of file base1_friend.h.

5.3.4.4 typedef struct base1_st_ base1_st

Friend accessible data for this class

5.3.4.5 typedef my_rc_e(∗ base1_string_fn)(base1_handle base1_h, char ∗buffer, size_t buffer_size)

Virtual function declaration.

Definition at line 59 of file base1_friend.h.

5.3.4.6 typedef my_rc_e(∗ base1_string_size_fn)(base1_handle base1_h, size_t ∗buffer_size)

Virtual function declaration.

Definition at line 65 of file base1_friend.h.

5.3.4.7 typedef const char∗(∗ base1_type_string_fn)(base1_handle base1_h)

Virtual function declaration.

Definition at line 53 of file base1_friend.h.

5.3.4.8 typedef struct base1_vtable_st_ base1_vtable_st

The virtual table to be specified by friend classes.

See also

base1_set_vtable()

5.3.5 Function Documentation

5.3.5.1 void base1_friend_delete (base1_handle *base1_h*)

Allow a friend class to delete the base1 object. It is assumed that the friend class is managing the memory for the base1 object and, thus, the object will not be freed. However, members within the base1 object may be freed. This does not call the virtual function table version of delete, but rather the delete specifically for type base1.

Parameters

base1_h	The object. If NULL, then this function is a no-op.

See also

base1_delete()

Definition at line 281 of file base1.c.

5.3.5.2 my_rc_e base1_inherit_vtable (const base1_vtable_st * *parent_vtable*, base1_vtable_st * *child_vtable*, bool *do_null_check*)

Fill in the child vtable with values inherited from the parent_vtable for all functions left NULL in the child vtable.

Parameters

parent_vtable	The parent vtable from which to inherit.
child_vtable	The child vtable to which functions may be inherited.
do_null_check	Indicates whether an error should be thrown if a function in the child vtable is NULL after inheritance.

Definition at line 382 of file base1.c.

5.3.5.3 my_rc_e base1_init (base1_handle *base1_h*)

Allows a friend class to initialize their inner base1 object. Must be called before the base1 object is used. If an error is returned, any clean-up was handled internally and there is no need to call a delete function.

Parameters

base1_h	The object

Returns

Return code

See also

base1_delete()
base1_friend_delete()

Definition at line 461 of file base1.c.

5.3.5.4 my_rc_e base1_set_vtable (base1_handle *base1_h*, base1_vtable_st * *vtable*)

This is a function used by friend classes to set the virtual table according to which methods they wish to override.

Parameters

base1_h	The object
vtable	The virtual table specification for the friend class. If a function pointer is NULL, then the base1 function is inherited.

Returns

Return code

Definition at line 425 of file base1.c.

5.4 base2.c File Reference

```
#include "base2_friend.h"
```

Classes

- struct base2_private_st_

Defines

- #define BASE2_STR_SIZE 64

Typedefs

- typedef struct base2_private_st_ base2_private_st

Functions

- my_rc_e base2_string_size (base2_handle base2_h, size_t *buffer_size)
- const char * base2_type_string (base2_handle base2_h)
- my_rc_e base2_string (base2_handle base2_h, char *buffer, size_t buffer_size)
- void base2_friend_delete (base2_handle base2_h)
- void base2_delete (base2_handle base2_h)
- my_rc_e base2_increase_val1 (base2_handle base2_h)
- my_rc_e base2_get_val1 (base2_handle base2_h, uint32_t *val1)
- my_rc_e base2_inherit_vtable (const base2_vtable_st *parent_vtable, base2_vtable_st *child_vtable, bool do_null_check)
- my_rc_e base2_set_vtable (base2_handle base2_h, base2_vtable_st *vtable)
- my_rc_e base2_init (base2_handle base2_h)

5.4.1 Detailed Description

Author

Matt Miller <matt@matthewjmiller.net>

5.4.2 LICENSE

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

5.4.3 DESCRIPTION

This is the implements a base class from which children class may inherit. Note that this is an abstract class with a pure virtual function and no constructor.

Definition in file base2.c.

5.4.4 Define Documentation

5.4.4.1 #define BASE2_STR_SIZE 64

Size for this object to use for base2_string_size_fn

Definition at line 29 of file base2.c.

5.4.5 Typedef Documentation

5.4.5.1 typedef struct base2_private_st_ base2_private_st

Private variables which cannot be directly accessed by any other class including children.

5.4.6 Function Documentation

5.4.6.1 void base2_delete (base2_handle *base2_h*)

Delete the object. This is a virtual function. Upon return, the object is not longer valid.

Parameters

base2_h	The object. If NULL, then this function is a no-op.

Definition at line 246 of file base2.c.

5.4.6.2 void base2_friend_delete (base2_handle *base2_h*)

Allow a friend class to delete the base2 object. It is assumed that the friend class is managing the memory for the base2 object and, thus, the object will not be freed. However, members within the base2 object may be freed. This does not call the virtual function table version of delete, but rather the delete specifically for type base2.

Parameters

base2_h	The object. If NULL, then this function is a no-op.

See also

base2_delete()

Definition at line 220 of file base2.c.

5.4.6.3 my_rc_e base2_get_val1 (base2_handle *base2_h,* uint32_t * *val1*)

Get the current val1 value for the object.

Parameters

base2_h	The object
val1	Outputs the current value.

Returns

Return code

Definition at line 285 of file base2.c.

5.4.6.4 my_rc_e base2_increase_val1 (base2_handle *base2_h*)

Increase val3 for the object. This is a pure virtual function.

Parameters

base2_h	The object

Returns

Return code

Definition at line 265 of file base2.c.

5.4.6.5 my_rc_e base2_inherit_vtable (const base2_vtable_st * *parent_vtable*, base2_vtable_st * *child_vtable*, bool *do_null_check*)

Fill in the child vtable with values inherited from the parent_vtable for all functions left NULL in the child vtable.

Parameters

parent_vtable	The parent vtable from which to inherit.
child_vtable	The child vtable to which functions may be inherited.
do_null_check	Indicates whether an error should be thrown if a function in the child vtable is NULL after inheritance.

Definition at line 320 of file base2.c.

5.4.6.6 my_rc_e base2_init (base2_handle *base2_h*)

Allows a friend class to initialize their inner base2 object. Must be called before the base2 object is used. If an error is returned, any clean-up was handled internally and there is no need to call a delete function.

Parameters

base2_h	The object

Returns

Return code

See also

base2_delete()
base2_friend_delete()

Definition at line 399 of file base2.c.

5.4.6.7 my_rc_e base2_set_vtable (base2_handle *base2_h*, base2_vtable_st * *vtable*)

This is a function used by friend classes to set the virtual table according to which methods they wish to override.

Parameters

base2_h	The object
vtable	The virtual table specification for the friend class. If a function pointer is NULL, then the base2 function is inherited.

Returns

Return code

Definition at line 363 of file base2.c.

5.4.6.8 my_rc_e base2_string (base2_handle *base2_h*, char * *buffer*, size_t *buffer_size*)

Get a string representation of the object. This is a virtual function.

Parameters

base2_h	The object
buffer	The buffer in which to put the string.
buffer_size	The size of the buffer.

Returns

Return code

See also

base2_string_size()

Definition at line 127 of file base2.c.

5.4.6.9 my_rc_e base2_string_size (base2_handle *base2_h*, size_t * *buffer_size*)

Get the minimum size of a string buffer that should be used to get a string representation of the object. This is a virtual function.

Parameters

base2_h	The object
buffer_size	Outputs the size of the buffer that should be used.

Returns

Return code

See also

base2_string()

Definition at line 50 of file base2.c.

5.4.6.10 const char* base2_type_string (base2_handle *base2_h*)

Get the string describing the type of the object. This is a virtual function.

Parameters

base2_h	The object

Returns

The string indicating the object type.

Definition at line 105 of file base2.c.

5.5 base2.h File Reference

```
#include "common.h"
```

Typedefs

- typedef struct base2_st_ * base2_handle

Functions

- my_rc_e base2_increase_val1 (base2_handle base2_h)
- my_rc_e base2_get_val1 (base2_handle base2_h, uint32_t *val1)
- void base2_delete (base2_handle base2_h)
- const char * base2_type_string (base2_handle base2_h)
- my_rc_e base2_string (base2_handle base2_h, char *buffer, size_t buffer_size)
- my_rc_e base2_string_size (base2_handle base2_h, size_t *buffer_size)

5.5.1 Detailed Description

Author

Matt Miller <matt@matthewjmiller.net>

5.5.2 LICENSE

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

5.5.3 DESCRIPTION

This is the public interface for base2 class.

Definition in file base2.h.

5.5.4 Typedef Documentation

5.5.4.1 typedef struct base2_st_* base2_handle

Opaque pointer to reference instances of this class

Definition at line 30 of file base2.h.

5.5.5 Function Documentation

5.5.5.1 void base2_delete (base2_handle *base2_h*)

Delete the object. This is a virtual function. Upon return, the object is not longer valid.

Parameters

base2_h	The object. If NULL, then this function is a no-op.

Definition at line 246 of file base2.c.

5.5.5.2 my_rc_e base2_get_val1 (base2_handle *base2_h,* uint32_t * *val1*)

Get the current val1 value for the object.

Parameters

base2_h	The object
val1	Outputs the current value.

Returns

Return code

Definition at line 285 of file base2.c.

5.5.5.3 my_rc_e base2_increase_val1 (base2_handle *base2_h*)

Increase val3 for the object. This is a pure virtual function.

Parameters

base2_h	The object

Returns

Return code

Definition at line 265 of file base2.c.

5.5.5.4 my_rc_e base2_string (base2_handle *base2_h*, char * *buffer*, size_t *buffer_size*)

Get a string representation of the object. This is a virtual function.

Parameters

base2_h	The object
buffer	The buffer in which to put the string.
buffer_size	The size of the buffer.

Returns

Return code

See also

base2_string_size()

Definition at line 127 of file base2.c.

5.5.5.5 my_rc_e base2_string_size (base2_handle *base2_h*, size_t * *buffer_size*)

Get the minimum size of a string buffer that should be used to get a string representation of the object. This is a virtual function.

Parameters

base2_h	The object
buffer_size	Outputs the size of the buffer that should be used.

Returns

Return code

See also

base2_string()

Definition at line 50 of file base2.c.

5.5.5.6 const char* base2_type_string (base2_handle *base2_h*)

Get the string describing the type of the object. This is a virtual function.

Parameters

base2_h	The object

Returns

The string indicating the object type.

Definition at line 105 of file base2.c.

5.6 base2_friend.h File Reference

```
#include "base2.h"
```

Classes

- struct base2_st_
- struct base2_vtable_st_

Typedefs

- typedef struct base2_private_st_ * base2_private_handle
- typedef struct base2_st_ base2_st
- typedef void(* base2_delete_fn)(base2_handle base2_h)
- typedef my_rc_e(* base2_increase_val1_fn)(base2_handle base2_h)
- typedef const char *(* base2_type_string_fn)(base2_handle base2_h)
- typedef my_rc_e(* base2_string_fn)(base2_handle base2_h, char *buffer, size_t buffer_size)
- typedef my_rc_e(* base2_string_size_fn)(base2_handle base2_h, size_t *buffer_size)
- typedef struct base2_vtable_st_ base2_vtable_st

Functions

- my_rc_e base2_inherit_vtable (const base2_vtable_st *parent_vtable, base2_vtable_st *child_vtable, bool do_null_check)
- my_rc_e base2_set_vtable (base2_handle base2_h, base2_vtable_st *vtable)
- void base2_friend_delete (base2_handle base2_h)
- my_rc_e base2_init (base2_handle base2_h)

5.6.1 Detailed Description

Author

Matt Miller <matt@matthewjmiller.net>

5.6.2 LICENSE

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

5.6.3 DESCRIPTION

This is the friend interface for base2 class. It should only be included by sub-classes of base2.

Definition in file base2_friend.h.

5.6.4 Typedef Documentation

5.6.4.1 typedef void(* base2_delete_fn)(base2_handle base2_h)

Virtual function declaration.

Definition at line 45 of file base2_friend.h.

5.6.4.2 typedef my_rc_e(* base2_increase_val1_fn)(base2_handle base2_h)

Virtual function declaration.

Definition at line 51 of file base2_friend.h.

5.6.4.3 typedef struct base2_private_st_* base2_private_handle

Opaque pointer to reference private data for the class

Definition at line 31 of file base2_friend.h.

5.6.4.4 typedef struct base2_st_ base2_st

Friend accessible data for this class

5.6.4.5 typedef my_rc_e(* base2_string_fn)(base2_handle base2_h, char *buffer, size_t buffer_size)

Virtual function declaration.

Definition at line 63 of file base2_friend.h.

5.6.4.6 typedef my_rc_e(* base2_string_size_fn)(base2_handle base2_h, size_t *buffer_size)

Virtual function declaration.

Definition at line 69 of file base2_friend.h.

5.6.4.7 typedef const char*(* base2_type_string_fn)(base2_handle base2_h)

Virtual function declaration.

Definition at line 57 of file base2_friend.h.

5.6.4.8 typedef struct base2_vtable_st_ base2_vtable_st

The virtual table to be specified by friend classes.

See also

base2_set_vtable()

5.6.5 Function Documentation

5.6.5.1 void base2_friend_delete (base2_handle *base2_h*)

Allow a friend class to delete the base2 object. It is assumed that the friend class is managing the memory for the base2 object and, thus, the object will not be freed. However, members within the base2 object may be freed. This does not call the virtual function table version of delete, but rather the delete specifically for type base2.

Parameters

base2_h	The object. If NULL, then this function is a no-op.

See also

base2_delete()

Definition at line 220 of file base2.c.

5.6.5.2 my_rc_e base2_inherit_vtable (const base2_vtable_st * *parent_vtable*, base2_vtable_st * *child_vtable*, bool *do_null_check*)

Fill in the child vtable with values inherited from the parent_vtable for all functions left NULL in the child vtable.

Parameters

parent_vtable	The parent vtable from which to inherit.
child_vtable	The child vtable to which functions may be inherited.
do_null_check	Indicates whether an error should be thrown if a function in the child vtable is NULL after inheritance.

Definition at line 320 of file base2.c.

5.6.5.3 my_rc_e base2_init (base2_handle *base2_h*)

Allows a friend class to initialize their inner base2 object. Must be called before the base2 object is used. If an error is returned, any clean-up was handled internally and there is no need to call a delete function.

Parameters

base2_h	The object

Returns

Return code

See also

base2_delete()
base2_friend_delete()

Definition at line 399 of file base2.c.

5.6.5.4 my_rc_e base2_set_vtable (base2_handle *base2_h*, base2_vtable_st * *vtable*)

This is a function used by friend classes to set the virtual table according to which methods they wish to override.

Parameters

base2_h	The object
vtable	The virtual table specification for the friend class. If a function pointer is NULL, then the base2 function is inherited.

Returns

Return code

Definition at line 363 of file base2.c.

5.7 common.c File Reference

```
#include "common.h"
```

Functions

- bool my_rc_e_is_notok (my_rc_e rc)
- bool my_rc_e_is_ok (my_rc_e rc)
- bool my_rc_e_is_valid (my_rc_e rc)
- const char * my_rc_e_get_string (my_rc_e rc)

5.7.1 Detailed Description

Author

Matt Miller <matt@matthewjmiller.net>

5.7.2 LICENSE

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

5.7.3 DESCRIPTION

This implements some common functions.

Definition in file common.c.

5.7.4 Function Documentation

5.7.4.1 const char* my_rc_e_get_string (my_rc_e *rc*)

Get a string representation of the return code.

Parameters

rc	The return code

Returns

A string representation of the return code or "__Invalid__" if an invalid return code is input.

Definition at line 88 of file common.c.

5.7.4.2 bool my_rc_e_is_notok (my_rc_e *rc*)

Indicates whether the return code is not in error.

Parameters

rc	The return code to check

Returns

true if the return code is not in error.

Definition at line 51 of file common.c.

5.7.4.3 bool my_rc_e_is_ok (my_rc_e *rc*)

Indicates whether the return code is in error.

Parameters

rc	The return code to check

Returns

true if the return code is not in error.

Definition at line 63 of file common.c.

5.7.4.4 bool my_rc_e_is_valid (my_rc_e *rc*)

Indicates whether the return code is valid.

Parameters

rc	The return code to check

Returns

true if the return code is valid.

Definition at line 75 of file common.c.

5.8 common.h File Reference

```
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include <stdbool.h>
#include <stddef.h>
#include <string.h>
```

Defines

- #define CT_ASSERT(e) extern char (∗CT_ASSERT(void)) [sizeof(char[1 - 2∗!(e)])]
- #define NELEMS(x) (sizeof(x) / sizeof(x[0]))
- #define LOG_ERR(fmt,...)
- #define VALIDATE_VTABLE_FN(obj_h, private_h, vtable, fn, rc)
- #define INHERIT_VTABLE_FN(parent_vtable, child_vtable, fn, do_null_check, rc)

Typedefs

- typedef enum my_rc_e_ my_rc_e

Enumerations

- enum my_rc_e_ {
  MY_RC_E_INVALID, MY_RC_E_SUCCESS, MY_RC_E_EINVAL, MY_RC_E_-ENOMEM,
  MY_RC_E_MAX }

Functions

- bool my_rc_e_is_notok (my_rc_e rc)
- bool my_rc_e_is_ok (my_rc_e rc)
- bool my_rc_e_is_valid (my_rc_e rc)
- const char ∗ my_rc_e_get_string (my_rc_e rc)

5.8.1 Detailed Description

Author

Matt Miller <matt@matthewjmiller.net>

5.8.2 LICENSE

5.8.3 DESCRIPTION

This is an interface for some common declarations.

Definition in file common.h.

5.8.4 Define Documentation

5.8.4.1 #define CT_ASSERT(e) extern char (*CT_ASSERT(void)) [sizeof(char[1 - 2*!(e)])]

Compile time assert macro from: http://www.pixelbeat.org/programming/gcc/static_assert.html

Definition at line 38 of file common.h.

5.8.4.2 #define INHERIT_VTABLE_FN(parent_vtable, child_vtable, fn, do_null_check, rc)

Value:

```
do { \
    if (NULL == child_vtable->fn) { \
        child_vtable->fn = parent_vtable->fn; \
        if (do_null_check && (NULL == child_vtable->fn)) { \
            LOG_ERR("Invalid input, " #fn "(%p)", child_vtable->fn); \
            rc = MY_RC_E_EINVAL; \
            goto err_exit; \
        } \
    } \
} while (0)
```

If the function in the child's table is NULL, then inherit the function from the parent table. In do_null_check is true, then it is considered an error condition if the function is NULL

after inheritance (e.g., when the base class is setting the vtable, all functions should be non-NULL). If the error condition is reached, we set the rc and goto an err_exit label.

Definition at line 97 of file common.h.

5.8.4.3 #define LOG_ERR(*fmt, ...*)

Value:

```
do { \
    printf("(%s:%u) ERROR: " fmt "\n", __FILE__, __LINE__, ##__VA_ARGS__); \
} while (0)
```

Display an error message.

Definition at line 48 of file common.h.

5.8.4.4 #define NELEMS(*x*) (sizeof(x) / sizeof(x[0]))

Determine the number of elements in an array.

Definition at line 43 of file common.h.

5.8.4.5 #define VALIDATE_VTABLE_FN(*obj_h, private_h, vtable, fn, rc*)

Value:

```
do { \
    if (NULL == obj_h) { \
        LOG_ERR("Invalid input, " #obj_h "(%p)", obj_h); \
        rc = MY_RC_E_EINVAL;  \
        break; \
    } \
\
    if (NULL == obj_h->private_h) { \
        LOG_ERR("Invalid input, " #obj_h "(%p) " #private_h "(%p)", obj_h, \
                obj_h->private_h); \
        rc = MY_RC_E_EINVAL;  \
        break; \
    } \
\
    if (NULL == obj_h->private_h->vtable) { \
        LOG_ERR("Invalid input, " #obj_h "(%p) " #private_h "(%p) " #vtable \
                "(%p)", obj_h, obj_h->private_h, obj_h->private_h->vtable); \
        rc = MY_RC_E_EINVAL;  \
        break; \
    } \
\
    if (NULL == obj_h->private_h->vtable->fn) { \
        LOG_ERR("Invalid input, " #obj_h "(%p) " #private_h "(%p) " #vtable \
                "(%p) " #fn "(%p)",  obj_h, obj_h->private_h, \
                obj_h->private_h->vtable, obj_h->private_h->vtable->fn); \
        rc = MY_RC_E_EINVAL;  \
        break; \
    } \
} while (0)
```

Validate that a function in an object's virtual table exists. If not, set the return code. Callers should set the return code to MY_RC_E_SUCCESS prior to calling the macro and check it after the macro is executed to make sure it is not an error return code.

Definition at line 59 of file common.h.

5.8.5 Typedef Documentation

5.8.5.1 typedef enum my_rc_e_ my_rc_e

Return codes used to indicate whether a function call was successful.

5.8.6 Enumeration Type Documentation

5.8.6.1 enum my_rc_e_

Return codes used to indicate whether a function call was successful.

Enumerator:

MY_RC_E_INVALID Invalid return code, should never be used

MY_RC_E_SUCCESS Successful return

MY_RC_E_EINVAL Function received an invalid input

MY_RC_E_ENOMEM Function failed to allocate memory

MY_RC_E_MAX Max return code for bounds testing

Definition at line 112 of file common.h.

5.8.7 Function Documentation

5.8.7.1 const char* my_rc_e_get_string (my_rc_e *rc*)

Get a string representation of the return code.

Parameters

rc	The return code

Returns

A string representation of the return code or "__Invalid__" if an invalid return code is input.

Definition at line 88 of file common.c.

5.8.7.2 bool my_rc_e_is_notok (my_rc_e *rc*)

Indicates whether the return code is not in error.

Parameters

rc	The return code to check

Returns

true if the return code is not in error.

Definition at line 51 of file common.c.

5.8.7.3 bool my_rc_e_is_ok (my_rc_e *rc*)

Indicates whether the return code is in error.

Parameters

rc	The return code to check

Returns

true if the return code is not in error.

Definition at line 63 of file common.c.

5.8.7.4 bool my_rc_e_is_valid (my_rc_e *rc*)

Indicates whether the return code is valid.

Parameters

rc	The return code to check

Returns

true if the return code is valid.

Definition at line 75 of file common.c.

5.9 derived1.c File Reference

```
#include "derived1_friend.h"
```

Classes

- struct derived1_private_st_

Defines

- #define DERIVED1_STR_SIZE 256

Typedefs

- typedef struct derived1_private_st_ derived1_private_st

Functions

- my_rc_e derived1_increase_val4 (derived1_handle derived1_h)
- void derived1_friend_delete (derived1_handle derived1_h)
- void derived1_delete (derived1_handle derived1_h)
- base1_handle derived1_cast_to_base1 (derived1_handle derived1_h)
- base2_handle derived1_cast_to_base2 (derived1_handle derived1_h)
- my_rc_e derived1_inherit_vtable (const derived1_vtable_st *parent_vtable, derived1_vtable_st *child_vtable, bool do_null_check)
- my_rc_e derived1_set_vtable (derived1_handle derived1_h, derived1_vtable_st *vtable)
- my_rc_e derived1_init (derived1_handle derived1_h)
- derived1_handle derived1_new1 (void)

5.9.1 Detailed Description

Author

Matt Miller <matt@matthewjmiller.net>

5.9.2 LICENSE

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

5.9.3 DESCRIPTION

This is the implements a class that inherits from base1 and base2.

Definition in file derived1.c.

5.9.4 Define Documentation

5.9.4.1 #define DERIVED1_STR_SIZE 256

Size for this object to use for base1_string_size_fn

Definition at line 27 of file derived1.c.

5.9.5 Typedef Documentation

5.9.5.1 typedef struct derived1_private_st_ derived1_private_st

Private variables which cannot be directly accessed by any other class including children.

5.9.6 Function Documentation

5.9.6.1 base1_handle derived1_cast_to_base1 (derived1_handle *derived1_h*)

Cast the derived1 object to base1.

Parameters

derived1_h	The derived1 object

Returns

The base1 object

Definition at line 449 of file derived1.c.

5.9.6.2 base2_handle derived1_cast_to_base2 (derived1_handle *derived1_h*)

Cast the derived1 object to base2.

Parameters

derived1_h	The derived1 object

Returns

The base2 object

Definition at line 467 of file derived1.c.

5.9.6.3 void derived1_delete (derived1_handle *derived1_h*)

Delete the object. This is a virtual function. Upon return, the object is not longer valid.

Parameters

derived1_h	The object. If NULL, then this function is a no-op.

Definition at line 368 of file derived1.c.

5.9.6.4 void derived1_friend_delete (derived1_handle *derived1_h*)

Allow a friend class to delete the derived1 object. It is assumed that the friend class is managing the memory for the derived1 object and, thus, the object will not be freed. However, members within the derived1 object may be freed. This does not call the virtual function table version of delete, but rather the delete specifically for type derived1.

Parameters

derived1_h	The object. If NULL, then this function is a no-op.

See also

derived1_delete()

Definition at line 342 of file derived1.c.

5.9.6.5 my_rc_e derived1_increase_val4 (derived1_handle *derived1_h*)

Increase val4 for the object. This is a virtual function.

Parameters

derived1_h	The object

Returns

Return code

Definition at line 288 of file derived1.c.

5.9.6.6 my_rc_e derived1_inherit_vtable (const derived1_vtable_st * *parent_vtable,* derived1_vtable_st * *child_vtable,* bool *do_null_check*)

Fill in the child vtable with values inherited from the parent_vtable for all functions left NULL in the child vtable.

Parameters

parent_vtable	The parent vtable from which to inherit.
child_vtable	The child vtable to which functions may be inherited.
do_null_check	Indicates whether an error should be thrown if a function in the child vtable is NULL after inheritance.

Definition at line 488 of file derived1.c.

5.9.6.7 my_rc_e derived1_init (derived1_handle *derived1_h*)

Allows a friend class to initialize their inner derived1 object. Must be called before the derived1 object is used. If an error is returned, any clean-up was handled internally and there is no need to call a delete function.

Parameters

derived1_h	The object

Returns

Return code

Definition at line 580 of file derived1.c.

5.9.6.8 derived1_handle derived1_new1 (void)

Create a new derived1 object.

Returns

The object or NULL if creation failed

Definition at line 648 of file derived1.c.

5.9.6.9 my_rc_e derived1_set_vtable (derived1_handle *derived1_h,* derived1_vtable_st * *vtable*)

This is a function used by friend classes to set the virtual table according to which methods they wish to override.

Parameters

derived1_h	The object
vtable	The virtual table specification for the friend class. If a function pointer is NULL, then the base1 function is inherited.

Returns

Return code

Definition at line 535 of file derived1.c.

5.10 derived1.h File Reference

```
#include "common.h"
#include "base1.h"
#include "base2.h"
```

Typedefs

- typedef struct derived1_st_ * derived1_handle

Functions

- my_rc_e derived1_increase_val4 (derived1_handle derived1_h)
- base1_handle derived1_cast_to_base1 (derived1_handle derived1_h)
- base2_handle derived1_cast_to_base2 (derived1_handle derived1_h)
- derived1_handle derived1_new1 (void)

5.10.1 Detailed Description

Author

Matt Miller <matt@matthewjmiller.net>

5.10.2 LICENSE

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

5.10.3 DESCRIPTION

This is the public interface for derived1 class, which inherits from base1 and base2.

Definition in file derived1.h.

5.10.4 Typedef Documentation

5.10.4.1 typedef struct derived1_st_* derived1_handle

Opaque pointer to reference instances of this class

Definition at line 33 of file derived1.h.

5.10.5 Function Documentation

5.10.5.1 base1_handle derived1_cast_to_base1 (derived1_handle *derived1_h*)

Cast the derived1 object to base1.

Parameters

derived1_h	The derived1 object

Returns

The base1 object

Definition at line 449 of file derived1.c.

5.10.5.2 base2_handle derived1_cast_to_base2 (derived1_handle *derived1_h*)

Cast the derived1 object to base2.

Parameters

derived1_h	The derived1 object

Returns

The base2 object

Definition at line 467 of file derived1.c.

5.10.5.3 my_rc_e derived1_increase_val4 (derived1_handle *derived1_h*)

Increase val4 for the object. This is a virtual function.

Parameters

derived1_h	The object

Returns

Return code

Definition at line 288 of file derived1.c.

5.10.5.4 derived1_handle derived1_new1 (void)

Create a new derived1 object.

Returns

The object or NULL if creation failed

Definition at line 648 of file derived1.c.

5.11 derived1_friend.h File Reference

```
#include "derived1.h"
#include "base1_friend.h"
#include "base2_friend.h"
```

Classes

- struct derived1_st_
- struct derived1_vtable_st_

Typedefs

- typedef struct derived1_private_st_ * derived1_private_handle
- typedef struct derived1_st_ derived1_st
- typedef void(* derived1_delete_fn)(derived1_handle derived1_h)
- typedef my_rc_e(* derived1_increase_val4_fn)(derived1_handle derived1_h)
- typedef struct derived1_vtable_st_ derived1_vtable_st

Functions

- my_rc_e derived1_inherit_vtable (const derived1_vtable_st *parent_vtable, derived1_vtable_st *child_vtable, bool do_null_check)
- my_rc_e derived1_set_vtable (derived1_handle derived1_h, derived1_vtable_st *vtable)
- void derived1_friend_delete (derived1_handle derived1_h)
- my_rc_e derived1_init (derived1_handle derived1_h)

5.11.1 Detailed Description

Author

Matt Miller <matt@matthewjmiller.net>

5.11.2 LICENSE

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

5.11.3 DESCRIPTION

This is the friend interface for derived1 class. It should only be included by sub-classes of derived1.

Definition in file derived1_friend.h.

5.11.4 Typedef Documentation

5.11.4.1 typedef void(∗ derived1_delete_fn)(derived1_handle derived1_h)

Virtual function declaration.

Definition at line 51 of file derived1_friend.h.

5.11.4.2 typedef my_rc_e(∗ derived1_increase_val4_fn)(derived1_handle derived1_h)

Virtual function declaration.

Definition at line 57 of file derived1_friend.h.

5.11.4.3 typedef struct derived1_private_st_∗ derived1_private_handle

Opaque pointer to reference private data for the class

Definition at line 33 of file derived1_friend.h.

5.11.4.4 typedef struct derived1_st_ derived1_st

Friend accessible data for this class

5.11.4.5 typedef struct derived1_vtable_st_ derived1_vtable_st

The virtual table to be specified by friend classes.

See also

derived1_set_vtable()

5.11.5 Function Documentation

5.11.5.1 void derived1_friend_delete (derived1_handle *derived1_h*)

Allow a friend class to delete the derived1 object. It is assumed that the friend class is managing the memory for the derived1 object and, thus, the object will not be freed. However, members within the derived1 object may be freed. This does not call the virtual function table version of delete, but rather the delete specifically for type derived1.

Parameters

derived1_h	The object. If NULL, then this function is a no-op.

See also

derived1_delete()

Definition at line 342 of file derived1.c.

5.11.5.2 my_rc_e derived1_inherit_vtable (const derived1_vtable_st * *parent_vtable*, derived1_vtable_st * *child_vtable*, bool *do_null_check*)

Fill in the child vtable with values inherited from the parent_vtable for all functions left NULL in the child vtable.

Parameters

parent_vtable	The parent vtable from which to inherit.
child_vtable	The child vtable to which functions may be inherited.
do_null_check	Indicates whether an error should be thrown if a function in the child vtable is NULL after inheritance.

Definition at line 488 of file derived1.c.

5.11.5.3 my_rc_e derived1_init (derived1_handle *derived1_h*)

Allows a friend class to initialize their inner derived1 object. Must be called before the derived1 object is used. If an error is returned, any clean-up was handled internally and there is no need to call a delete function.

Parameters

derived1_h	The object

Returns

Return code

Definition at line 580 of file derived1.c.

5.11.5.4 my_rc_e derived1_set_vtable (derived1_handle *derived1_h*, derived1_vtable_st * *vtable*)

This is a function used by friend classes to set the virtual table according to which methods they wish to override.

Parameters

derived1_h	The object
vtable	The virtual table specification for the friend class. If a function pointer is NULL, then the base1 function is inherited.

Returns

Return code

Definition at line 535 of file derived1.c.

5.12 derived2.c File Reference

```
#include "derived2.h"
#include "derived1_friend.h"
```

Classes

- struct derived2_st_

Defines

- #define DERIVED2_STR_SIZE 256

Typedefs

- typedef struct derived2_st_ derived2_st

Functions

- derived1_handle derived2_cast_to_derived1 (derived2_handle derived2_h)
- derived2_handle derived2_new1 (void)

5.12.1 Detailed Description

Author

Matt Miller <matt@matthewjmiller.net>

5.12.2 LICENSE

5.12.3 DESCRIPTION

This is the implements a class that inherits from derived1.

Definition in file derived2.c.

5.12.4 Define Documentation

5.12.4.1 #define DERIVED2_STR_SIZE 256

Size for this object to use for base1_string_size_fn

Definition at line 28 of file derived2.c.

5.12.5 Typedef Documentation

5.12.5.1 typedef struct derived2_st_ derived2_st

Private data for this class

5.12.6 Function Documentation

5.12.6.1 derived1_handle derived2_cast_to_derived1 (derived2_handle *derived2_h*)

Cast the derived2 object to derived1.

Parameters

derived2_h	The derived2 object

Returns

The derived1 object

Definition at line 252 of file derived2.c.

5.12.6.2 derived2_handle derived2_new1 (void)

Create a new derived2 object.

Returns

The object or NULL if creation failed

Definition at line 302 of file derived2.c.

5.13 derived2.h File Reference

```
#include "common.h"
#include "derived1.h"
```

Typedefs

- typedef struct derived2_st_ * derived2_handle

Functions

- derived1_handle derived2_cast_to_derived1 (derived2_handle derived2_h)
- derived2_handle derived2_new1 (void)

5.13.1 Detailed Description

Author

Matt Miller <matt@matthewjmiller.net>

5.13.2 LICENSE

5.13.3 DESCRIPTION

This is the public interface for derived2 class, which inherits from derived2.

Definition in file derived2.h.

5.13.4 Typedef Documentation

5.13.4.1 typedef struct derived2_st_* derived2_handle

Opaque pointer to reference instances of this class

Definition at line 32 of file derived2.h.

5.13.5 Function Documentation

5.13.5.1 derived1_handle derived2_cast_to_derived1 (derived2_handle *derived2_h*)

Cast the derived2 object to derived1.

Parameters

derived2_h	The derived2 object

Returns

The derived1 object

Definition at line 252 of file derived2.c.

5.13.5.2 derived2_handle derived2_new1 (void)

Create a new derived2 object.

Returns

The object or NULL if creation failed

Definition at line 302 of file derived2.c.

5.14 test_c_oo.c File Reference

```
#include "base1.h"
#include "base2.h"
#include "derived1.h"
#include "derived2.h"
```

Functions

- int main (int argc, char *argv[])

5.14.1 Detailed Description

Author

Matt Miller <matt@matthewjmiller.net>

5.14.2 LICENSE

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

5.14.3 DESCRIPTION

Main program to do some basic tests on the object-oriented C code.

Definition in file test_c_oo.c.

5.14.4 Function Documentation

5.14.4.1 int main (int *argc,* char * *argv[]*)

Main function to test objects.

Definition at line 131 of file test_c_oo.c.

Index