NAME

sptr, **sptr2**, **sptr3**, **sptr_free**, **sptr_dup**, **sptr_resize**, **sptr_size**, **sptr_ref**, **sptr_cpy**, **sptr_rel**, **sptr_array**, **sptr_array2**, **sptr_array_resize**, **sptr_array_size**, **sptr_del_ptr**, **sptr_del_sptr**, **sptr_del_array**, **sptr_del_ptr_array**, **sptr_del_sptr_array** — smart pointer functionality

LIBRARY

Smart Pointer Library for C (libsptr, -lsptr)

SYNOPSIS

```
#include <sptr.h>

typedef void (* sptr_del_fn) (void * value, size_t value_size,
void * metadata, size_t metadata_size);

void *
sptr(size_t size, init_value, sptr_del_fn del_value, U metadata);

void *
sptr2(T value, sptr_del_fn del_value, U metadata);

void *
sptr3(size_t ptr_size, void * ptr, sptr_del_fn del_value, U metadata);

void
sptr_free(void * ptr);

void *
sptr_dup(void * ptr);

bool
sptr_resize(void * ptr, size_t new_size);

size_t
sptr_size(void * ptr);

unsigned int
sptr_ref(void * ptr);

void
sptr_cpy(void * ptr);

void
sptr_rel(void * ptr);

#include <sptr/util.h>

typedef void (* sptr_del_element_fn) (void * element);

void *
sptr_array(element_type, size_t count, init_value,
    sptr_del_element_fn del_element);

void *
sptr_array2(element_type, T[] array, sptr_del_element_fn del_element);

bool
sptr_array_resize(void * ptr, size_t new_count);

size_t
sptr_array_size(void * ptr);

void
sptr_del_ptr(void * ptr, size_t value_size, void * metadata,
    size_t metadata_size);
```

void
sptr_del_sptr(*void * ptr*, *size_t value_size*, *void * metadata*,
size_t metadata_size);

void
sptr_del_array(*void * ptr*, *size_t value_size*, *void * metadata*,
size_t metadata_size);

void
sptr_del_ptr_array(*void * ptr*, *size_t value_size*, *void * metadata*,
size_t metadata_size);

void
sptr_del_sptr_array(*void * ptr*, *size_t value_size*, *void * metadata*,
size_t metadata_size);

DESCRIPTION

Smart pointers have the same types as their corresponding ordinary pointers in the C language, but include extra information that enables certain "smart" behaviour.

- When a smart pointer is created, a destructor may be specified, which is called when the smart pointer is freed.
- Smart pointers are reference-counted; they are automatically freed when the reference count reaches zero.
- Any smart pointer variable tagged with the *smart* attribute is automatically released when its scope is exited.
- Smart pointers can be manually copied, released, resized, and freed.
- Smart pointers are thread-safe and all operations on them are lock-free.

The core functionality for smart pointers is defined entirely in <sbuf.h>, and does not strictly require linking to the library.

Note that any smart pointer that should be automatically released needs to be declared with the *smart* attribute (before the type name of the declaration).

The **sptr**(), **sptr2**(), and **sptr3**() macros create a new smart pointer. **sptr**() creates a pointer referencing a block of memory of size *size* bytes. The array may be initialized by specifying an arbitrary expression (constructor) to the *init_value* parameter, where *void * value* is a pointer to the block of memory (the same pointer that is returned); the macro noop should be specified if no initialization is desired. **sptr2**() creates a pointer referencing a block of memory that initially contains the value of the expression *value*, which can be of any non-void type. **sptr3**() creates a pointer copied from the value at location *ptr*, of size *size* bytes. For all of **sptr**(), **sptr2**(), and **sptr3**(), the argument *del_value* specifies the function of type *sptr_del_fn* to call when the pointer is freed using **sptr_free**(). Arbitrary metadata is stored along with the pointer by passing an expression to the *metadata* parameter.

The **sptr_free**() macro frees the block of memory allocated for the smart pointer *ptr*.

The **sptr_dup**() macro takes an existing smart pointer *ptr* and creates a duplicate smart pointer.

The **sptr_resize**() macro resizes the block of memory referenced by the smart pointer *ptr* to *new_size* bytes.

The **sptr_size**() macro gets the size in bytes of the block of memory allocated for the smart pointer *ptr*.

The **sptr_ref**() macro gets the number of active references to the smart pointer *ptr*.

The **sptr_cpy**() macro makes a new reference to the smart pointer *ptr*, incrementing the reference count by one.

The **sptr_rel**() macro releases a reference to the smart pointer *ptr*, decrementing the reference count by one. When the reference count reaches zero, **sptr_free**() is automatically performed on the pointer.

Utilities relating to smart pointers are defined in <sbuf.h>, and requires linking to the library.

The **sptr_array**() and **sptr_array2**() macros create a new smart pointer for storing an array. For both macros, the argument *element_type* specifies the type of the elements of the array. **sptr_array**() creates a pointer referencing a block of memory of sufficient size to store *count* objects of type *element_type*. The array may be initialized by specifying an arbitrary expression (constructor) to the *init_value* parameter, as for the **sptr**() macro. **sptr_array2**() creates a pointer for storing the contents of the array *array*. For both **sptr_array**() and **sptr_array2**(), the argument *del_element* specifies the function of type *sptr_del_element_fn* to call for each element of the array when the pointer is freed using **sptr_free**().

The **sptr_array_resize**() macro resizes the block of memory referenced by the array smart pointer *ptr* to *new_size* objects of the element type.

The **sptr_array_size**() macro gets the size of the array stored by the block of memory allocated for the array smart pointer *ptr*.

The functions **sptr_del_ptr**(), **sptr_del_sptr**(), **sptr_del_array**(), **sptr_del_ptr_array**(), and **sptr_del_sptr_array**() are all smart pointer destructors (of type *sptr_del_fn*), and are intended to be passed as the *del_value* argument to **sptr**(), **sptr2**(), and **sptr3**() or composed into other user-defined destructors.

RETURN VALUES

The **sptr**(), **sptr2**(), and **sptr3**() macros return the created smart pointer, or NULL if the creation failed.

The **sptr_dup**() macro returns the duplicate smart pointer, or NULL if the duplication failed.

The **sptr_resize**() macro returns true if the smart pointer was resized, or false if the resize operation failed.

The **sptr_size**() macro returns the size in bytes of the block of memory referenced by the smart pointer.

The **sptr_ref**() macro returns the number of active references to the smart pointer.

The **sptr_array**() and **sptr_array2**() macros return the created smart pointers, or NULL if the creation failed.

The **sptr_array_size**() macro returns the size of the array referenced by the smart pointer.

ERRORS

The **sptr**(), **sptr2**(), **sptr**(*3*), **sptr_dup**(), **sptr_array**(), and **sptr_array2**() macros may fail and set *errno* for any of the errors specified for the routine malloc(3).

The **sptr_resize**() and **sptr_array_resize**() macros may fail and set *errno* for any of the errors specified for the routine realloc(3).

The **sptr_free**() and **sptr_rel**() macros may fail and set *errno* for any of the errors specified for the routine free(3).

SEE ALSO

free(3), malloc(3), realloc(3)

AUTHORS

Alexander Regueiro ⟨alex@noldorin.com⟩