Memory Management Function Replacements of the C and C++ Standard Libraries: Tutorial

PlayStation®Vita Programmer Tool Runtime Library Release 3.50

Document serial number: 000004892117

Table of Contents

1 Overview 3

About This Tutorial 3

Procedure for Replacing the Memory Management Functions 3

Sample Program 3

2 Sample Program 4

Memory Management Function Replacement on PlayStation®Vita 4

Replacement of the C Language Library Functions 4

Replacement of the C Language Library Functions (for TLS) 5

Replacement of the C++ Language Library Operators 5

1 Overview

About This Tutorial

This tutorial explains, if it is necessary to replace the memory management functions of the C and C++ standard libraries on PlayStation®Vita, how to replace them in the supported way.

Procedure for Replacing the Memory Management Functions

In PlayStation®Vita, the memory management function of the C/C++ standard library can be replaced with the following procedure.

(1) Create an object file with the necessary functions defined (e.g., `user_malloc`/`user_free`)

(2) Explicitly link the object file to the main module

(3) Upon execution of an application, the replacement will take place when the C and C++ standard library module is initialized

Replacing the memory management function according to the above procedure, in the application process, all functions called by the memory management function become the memory management functions defined for replacement. In particular, a call of the memory management function from locations indicated below is replaced.

- Main module with an object linked directly
- The C/C++ standard library (`libc.suprx`)
- PRX using the memory management function of the C/C++ standard library

Note that if the memory management function is just defined directly in the main module, the defined memory management function is only used in the main module and is not used in the C/C++ standard library or other PRX.

Sample Program

A sample program of this tutorial is provided in the following directory.

- sample_code/system/tutorial_malloc_replace/

2 Sample Program

In the tutorial_malloc_replace sample program, the method for replacing memory management functions is used to replace the memory management functions of the C and C++ standard libraries with user-defined memory management functions.

The sample program uses a system call to allocate a memory area of 1MiB and uses this as heap area using `mspace`. Functions, such as `mspace_malloc()` of the C library, are used in the algorithm to allocate memory from the heap area. In addition, to confirm the replacement, a message is output within the defined functions.

The sample program is composed of the following files.

main.cpp	Test code to check the replacement
user_malloc.c	Definitions of replacement functions of the C language library
user_malloc_for_tls.c	Definitions of replacement functions of the C language library (for TLS)
user_new.cpp	Definitions of replacement functions of the C++ language library

Memory Management Function Replacement on PlayStation®Vita

(1) Create an object file with the necessary functions defined.

(2) Explicitly link the object file to the main module.

(3) Upon execution, the replacement will take place when the C and C++ standard library module is initialized.

When an object file, which lacks the required function definitions, is linked to the main module, a trap instruction will stop the processing upon initialization of the C and C++ standard library module.

Replacement of the C Language Library Functions

To replace the following C language library functions,

- `void malloc(size_t size)`
- `void free(void *ptr)`
- `void *calloc(size_t nelem, size_t size)`
- `void *realloc(void *ptr, size_t size)`
- `void *memalign(size_t boundary, size_t size)`
- `void *reallocalign(void *ptr, size_t size, size_t boundary)`
- `int malloc_stats(struct malloc_managed_size *mmsize)`
- `int malloc_stats_fast(struct malloc_managed_size *mmsize)`
- `size_t malloc_usable_size(void *ptr)`

an object file must be created with the following functions defined according to the “Memory Management Function Replacements of the C and C++ Standard Libraries: Reference” document.

- `void user_malloc_init(void)`
- `void user_malloc_finalize(void)`
- `void *user_malloc(size_t size)`
- `void user_free(void *ptr)`
- `void *user_calloc(size_t nelem, size_t size)`
- `void *user_realloc(void *ptr, size_t size)`
- `void *user_memalign(size_t boundary, size_t size)`
- `void *user_reallocalign(void *ptr, size_t size, size_t boundary)`

- `int user_malloc_stats(struct malloc_managed_size *mmsize)`
- `int user_malloc_stats_fast(struct malloc_managed_size *mmsize)`
- `size_t user_malloc_usable_size(void *ptr)`

These functions are defined in user_malloc.c of the sample program.

These functions must all be defined whether they are used or not.

These functions must be defined as C language functions. If you define them in the C++ language, define them with `extern "C"`.

Replacement of the C Language Library Functions (for TLS)

Since SDK 1.800, the implementation of TLS became to use the memory management functions. Therefore, if the functions defined for the replacement of the C language library functions use TLS, they will not work correctly. In such a case, replace the C language library functions for TLS with user-defined memory management functions. To replace them, an object file must be created with the following functions defined according to the "Memory Management Function Replacements of the C and C++ Standard Libraries: Reference" document.

- `void user_malloc_for_tls_init(void)`
- `void user_malloc_for_tls_finalize(void)`
- `void *user_malloc_for_tls(size_t size)`
- `void user_free_for_tls(void *ptr)`

These functions are defined in user_malloc_for_tls.c of the sample program.

These functions must all be defined.

These functions must be defined as C language functions. If you define them in the C++ language, define them with `extern "C"`.

Even if memory management functions for TLS are replaced, `user_malloc_init()` cannot use TLS.

The replaced functions will be executed in the following order at the initialization.

(1) `user_malloc_for_tls_init()`
(2) `user_malloc_init()`
(3) Initialization of TLS of the main module
(4) Global constructors of the main module
(5) `main()`

Replacement of the C++ Language Library Operators

To replace the following C++ language library operators,

- `operator new(std::size_t size) throw(std::bad_alloc)`
- `operator new(std::size_t size, const std::nothrow_t&) throw()`
- `operator new[](std::size_t size) throw(std::bad_alloc)`
- `operator new[](std::size_t size, const std::nothrow_t&) throw()`
- `operator delete(void *ptr) throw()`
- `operator delete(void *ptr, const std::nothrow_t&) throw()`
- `operator delete[](void *ptr) throw()`
- `operator delete[](void *ptr, const std::nothrow_t&) throw()`

an object file must be created with the following functions defined according to the "Memory Management Function Replacements of the C and C++ Standard Libraries: Reference" document.

- `void *user_new(std::size_t size) throw(std::bad_alloc)`

- `void *user_new(std::size_t size, const std::nothrow_t&) throw()`
- `void *user_new_array(std::size_t size) throw(std::bad_alloc)`
- `void *user_new_array(std::size_t size, const std::nothrow_t&) throw()`
- `void user_delete(void *ptr) throw()`
- `void user_delete(void *ptr, const std::nothrow_t&) throw()`
- `void user_delete_array(void *ptr) throw()`
- `void user_delete_array(void *ptr, const std::nothrow_t&) throw()`

These functions are defined in user_new.cpp of the sample program.

These functions must all be defined whether they are used or not.

When the C language library functions are replaced without replacing the C++ language library operators, the standard operators of the C++ language library will use the replaced memory management functions.