z/OS V2.5 IBM Education Assistant

Solution Name: LE support for C11 C/C++ standards items – aligned_alloc() (Part I)

Solution Element(s): z/OS LE

Agenda

- Trademarks
- Objectives
- Overview
- Usage & Invocation
- Interactions & Dependencies
- Upgrade & Coexistence Considerations
- Installation & Configuration
- Summary
- Appendix

Trademarks

- See url http://www.ibm.com/legal/copytrade.shtml for a list of trademarks.
- Additional Trademarks:
  - None

Objectives

Introduction to LE support for C11 C/C++ standards items – aligned_alloc() (Part I).

Overview

- Who (Audience)

  As an application programmer, I want C11 aligned_alloc() support available, so I can develop/port standardized and modernized application more easily.

- What (Solution)

  This is the first part support for aligned_alloc(). In this part, a new function aligned_alloc() will be added with the following limitations:

  - For alignment

    AMODE 31 only supports up to 8 bytes alignment

    AMODE 64 only supports up to 16 bytes alignment.

- Wow (Benefit / Value, Need Addressed)

  Application user can develop/port standardized and modernized application more easily by invoking aligned_alloc().

Usage & Invocation

This is the first part support for aligned_alloc(). In this part, a new function aligned_alloc() will be added with the following limitations:

For alignment

AMODE 31 only supports up to 8 bytes alignment

AMODE 64 only supports up to 16 bytes alignment.

- **Format**

  #include <stdlib.h>
  void *aligned_alloc(size_t alignment, size_t size);

- **General Description**

  The aligned_alloc function allocates space for an object whose alignment is specified by alignment, whose size is specified by size, and whose value is indeterminate.

- **Parameter description**

  alignment - Specifies the alignment, *alignment* is a power of two.

  Note: For alignment,AMODE 31 only supports up to 8 bytes alignment, AMODE 64 only supports up to 16 bytes alignment.

  size - Number of bytes to allocate. An integral multiple of alignment.

Usage & Invocation

- **Usage notes**

  To use the aligned_alloc() function, compile the source code with the LANGLVL(EXTC1X) option.

- ***Returned Value***

  On success, returns the pointer to the beginning of newly allocated memory. To avoid a memory leak, the returned pointer must be deallocated with free() or realloc().

  On failure, returns a null pointer, it sets errno to one of the following values:

  Error Code Description

  EINVAL

  The value of *alignment* is not supported.

  ENOMEM

  Insufficient memory is available

Usage & Invocation

Test Case example:

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>

int main(void)
{
int *p1 = aligned_alloc(8, 8*sizeof *p1);
if(!p1)
{
printf("aligned_alloc failed with errno = %d\n", errno);
return -1;
}
else
{
   printf("8-byte aligned addr: %p\n", (void*)p1);
      free(p1);
}
return 0;
}
```

Possible output:

8-byte aligned addr: 22575D20

Interactions & Dependencies

- Software Dependencies
  - z/OS LE V2R5
- Hardware Dependencies
  - None
- Exploiters
  - None

Upgrade & Coexistence Considerations

- To exploit this solution, all systems in the Sysplex must be at the new z/OS level: No

Installation & Configuration

- None

Summary

Previously, z/OS LE does not Support for C11 C/C++ standards items – aligned_alloc() .

The restrictions are removed on V2R5.

Appendix

- z/OS XL C/C++ Runtime Library Reference