Capability space management library

Generated by Doxygen 1.8.1.2

Wed Jul 31 2013 13:33:04

Contents

1 Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

cspace_t
A representation of a cspace_t **1**

2 File Index

2.1 File List

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

cspace.h
A Cspace management library **2**

3 Data Structure Documentation

3.1 cspace_t Struct Reference

A representation of a cspace_t.

Data Fields

- seL4_CPtr root_cnode

A seL4_CPtr to the root cnode of this cspace.

- seL4_CapData_t guard

  The seL4_CapData representing the guard needed by a TCB to use this cspace.

3.1.1 Detailed Description

A representation of a cspace_t.

The Internals Should be opaque to users except for the two variables documented, which a needed to initialise TCBs: root_cnode and guard.

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

- cspace.h

4 File Documentation

4.1 cspace.h File Reference

A Cspace management library.

Data Structures

- struct cspace_t

  A representation of a cspace_t.

Macros

- #define CSPACE_NODE_SIZE_IN_MEM_BITS seL4_PageDirBits

  Defines the cnode size to be the same as a page directory (16K on ARM) in 'bits'.
- #define CSPACE_NODE_SIZE_IN_MEM (1 << CSPACE_NODE_SIZE_IN_MEM_BITS)

  The cnode size in bytes.
- #define CSPACE_NODE_SIZE_IN_SLOTS_BITS (CSPACE_NODE_SIZE_IN_MEM_BITS - seL4_SlotBits)

  The cnode size in term of the number of slots (in bits). Used by seL4 operations.
- #define CSPACE_NODE_SIZE_IN_SLOTS (1 << CSPACE_NODE_SIZE_IN_SLOTS_BITS)

  The cnode size in actual number of slots.
- #define CSPACE_NULL 0

  Functions that return a seL4_CPtr return CSPACE_NULL if they fail.
- #define CSPACE_DEPTH 32

  Cspace depth of cspaces managed by this library.

Typedefs

- typedef seL4_Word(∗ cspace_ut_alloc_t)(int size_in_bits)

  The type of a function pointer to an untyped memory allocator.
- typedef void(∗ cspace_ut_free_t)(seL4_Word address, int size_in_bits)

  The corresponding free function pointer type for cspace_ut_alloc_t.
- typedef int(∗ cspace_ut_translate_t)(seL4_Word address, seL4_CPtr ∗ut_cptr, seL4_Word ∗offset)

  The function pointer type to translate a physical address into an untyped capability and an offset within it.
- typedef void ∗(∗ cspace_malloc_t)(size_t bytes)

  A function pointer type of a malloc equivalent.
- typedef void(∗ cspace_free_t)(void ∗addr)

  The obvious complement to cspace_malloc_t.

Enumerations

- enum cspace_err_t

  The error type for cspace functions that don't return a seL4_CPtr.

Functions

- cspace_err_t cspace_root_task_bootstrap (cspace_ut_alloc_t ut_alloc, cspace_ut_free_t ut_free, cspace_ut-_translate_t ut_translate, cspace_malloc_t malloc, cspace_free_t free)

  Bootstrap the initial tasks cspace from what seL4 provides initially.
- cspace_t ∗ cspace_create (int levels)

  Allocate a new cspace based on the specified number of levels.
- cspace_err_t cspace_destroy (cspace_t ∗c)

  Destroy the specified cspace and return it's resources.
- seL4_CPtr cspace_alloc_slot (cspace_t ∗c)

  Reserve a free slot in the specified cspace.
- cspace_err_t cspace_free_slot (cspace_t ∗c, seL4_CPtr slot)

  Return a slot back to the cspace slot allocator.
- seL4_CPtr cspace_copy_cap (cspace_t ∗dest, cspace_t ∗src, seL4_CPtr src_cap, seL4_CapRights rights)

  Copy a capability from one cspace to the same or another cspace.
- cspace_err_t cspace_delete_cap (cspace_t ∗c, seL4_CPtr cap)

  Delete a specified capability.
- seL4_CPtr cspace_mint_cap (cspace_t ∗dest, cspace_t ∗src, seL4_CPtr src_cap, seL4_CapRights rights, seL4_CapData_t badge)

  Mint a new capability into the specified cspace.
- seL4_CPtr cspace_move_cap (cspace_t ∗dest, cspace_t ∗src, seL4_CPtr src_cap)

  Move a cap from a specified location to another location.
- seL4_CPtr cspace_mutate_cap (cspace_t ∗dest, cspace_t ∗src, seL4_CPtr src_cap, seL4_CapData_-t badge)

  Move a capability and set it's badge in the process.
- cspace_err_t cspace_recycle_cap (cspace_t ∗c, seL4_CPtr cap)

  Recycle the specified capability.
- cspace_err_t cspace_revoke_cap (cspace_t ∗c, seL4_CPtr cap)

  Attempt to revoke all capabilities derived from the specified capability.
- seL4_CPtr cspace_rotate_cap (cspace_t ∗dest, seL4_CapData_t dest_badge, cspace_t ∗pivot, seL4_CPtr pivot_cap, seL4_CapData_t pivot_badge, cspace_t ∗src, seL4_CPtr src_cap)

  Rotate two caps between three slots.
- seL4_CPtr cspace_save_reply_cap (cspace_t ∗dest)

  Save the callers reply capability.
- seL4_CPtr cspace_irq_control_get_cap (cspace_t ∗dest, seL4_IRQControl irq_cap, int irq)

  Create an IRQ handler capability in the specified cspace.
- seL4_Error cspace_ut_retype_addr (seL4_Word address, seL4_Word type, seL4_Word size_bits, cspace_t ∗dest, seL4_CPtr ∗dest_cap)

  Create a new kernel object from untyped memory.

Variables

- cspace_t ∗ cur_cspace

  A globally visible for referring to the root tasks current cspace.

4.1.1 Detailed Description

A Cspace management library.

Date

Wed Jul 18 03:48:06 2012

The library attempts to abstract away the management of seL4 Cnodes that store capabilities. Cnodes are nodes of capability storage that are combined to form Cspaces. A Cspace is simply an indexable capability address space.

Specifically, the library allocates and de-allocates cnodes as required and combines them to form a capability address space indexable by a seL4_CPtr. The library supports 1-level and 2-level cspaces in a in an analogous way to 1-level and 2-level page tables. However, except for the maximum potential cspace index, the underlying number of levels is hidden from the interface.

The library also takes care of free slot management in the cspace. seL4 cnode operations that would normally require a free slot as an argument are wrapped by this library such that only specifying the destination cspace is required, with the library managing the free slot allocation internally. Likewise, seL4 operations that have arguments that require a detailed understanding of the cnode layout (e.g. the depth argument to some operations) are also wrapped by the library to avoid the library user needing to understand the library internals completely.

Note: A one-level cspace's index ranges from 1..(CSPACE_NODE_SIZE_IN_SLOTS - 1), and a two-level cspace ranges from 1..(CSPACE_NODE_SIZE_IN_SLOTS ∗ CSPACE_NODE_SIZE_IN_SLOTS - 1).

4.1.2 Typedef Documentation

4.1.2.1 typedef seL4_Word(∗ cspace_ut_alloc_t)(int size_in_bits)

The type of a function pointer to an untyped memory allocator.

Parameters

size_in_bits	The size of the requested object in memory in bits (i.e. Log2 actual size).

Returns

Return NULL if it fails.

This function pointer gets set by the bootstrapping code from a function pointer passed in from the external environment.

The function is expected to allocate regions of untyped memory of the specified size for the cnodes allocated internally by this library. The allocator is also expected to align the object to it's size.

Note: The function provided must itself not result in a call back into this cspace library as this might result in deadlock or infinite recursion.

Also Note: This function is intended to track the allocation of untyped memory. However, it is a user-level library, and as such, its state can diverge from the actual kernel state which will result in attempting to allocate overlapping kernel object, which seL4 will deny. Like all allocators, care should be taken to ensure its state reflects that of the kernel.

4.1.2.2 typedef void(∗ cspace_ut_free_t)(seL4_Word address, int size_in_bits)

The corresponding free function pointer type for cspace_ut_alloc_t.

Parameters

address	The physical address to free.
size_in_bits	The size in bits of the object to be freed.

Returns

Always succeeds.

4.1.2.3 typedef int(∗ cspace_ut_translate_t)(seL4_Word address, seL4_CPtr ∗ut_cptr, seL4_Word ∗offset)

The function pointer type to translate a physical address into an untyped capability and an offset within it.

Parameters

address	The physical memory address.
ut_cptr	The returned seL4_CPtr to the untyped capability.
offset	The returned offset within the untyped.

Returns

0 on success

This function pointer is provided by the untyped allocator. It must take the address originally provided by the allocator itself and convert the address into an (ut_cptr,offset) tuple. The exact form of the address and how the translation is performed is up to the allocator.

Note: This function must not cause an indirect call back into the cspace library.

4.1.2.4 typedef void∗(∗ cspace_malloc_t)(size_t bytes)

A function pointer type of a malloc equivalent.

Parameters

bytes	The malloc equivalent used internal to the cspace library. It is passed in at bootstrap and used from then on. The cspace library does not use the normal malloc directly as this malloc must not indirectly result in calling back into the cspace library. For the time being the default malloc adheres to this requirement, but we don't expect that to always be true.

4.1.2.5 typedef void(∗ cspace_free_t)(void ∗addr)

The obvious complement to cspace_malloc_t.

Parameters

addr	

4.1.3 Function Documentation

4.1.3.1 cspace_err_t cspace_root_task_bootstrap (cspace_ut_alloc_t *ut_alloc,* cspace_ut_free_t *ut_free,* cspace_ut_translate_t *ut_translate,* cspace_malloc_t *malloc,* cspace_free_t *free*)

Bootstrap the initial tasks cspace from what seL4 provides initially.

Parameters

ut_alloc	An allocation function that the library will use to allocate cnodes, i.e. the physical address of free untyped kernel memory. Note: The library assumes that this function will not call back into the library itself – things are likely to break if it does.
ut_free	The corresponding free function for untyped memory.
ut_translate	A function that takes a physical address in untyped memory and returns the corresponds untyped capability and offset.
malloc	A function that implements malloc like functionality, i.e. real accessible memory.
free	The corresponding free function for malloc.

Returns

Either CSPACE_ERROR or CSPACE_NOERROR

This routine takes the root task's initial cnode configuration it gets from seL4 and transforms it into the two-level cspace layout that this library can manipulate.

It involves:

- Initialising function pointer to external allocators. See typedef for details of the requirements of those functions.
- Allocating a cspace struct initialised appropriately
- Creating a two level cspace sufficient to hold the content of the boot cnode.
- Moving the initial caps across to so as to preserve bootinfo layout (i.e. indexes continue to work).
- Removing the initial cnode.

4.1.3.2 cspace_t* cspace_create (int *levels*)

Allocate a new cspace based on the specified number of levels.

Parameters

levels	The number of desired levels in the new cspace. The library only support 1 or 2 levels.

Returns

On failure it returns NULL

Return a pointer the some book keeping that keeps track of allocated cnodes with the new cspace, and which slots are free etc... The intention is not to expose the internal cspace specifics, only to expose seL4_CPtr's valid in the specific cspace_t.

The cspace is initialised such that there are free slots upon return. I.e. 1 or 2 cnodes are allocated depending of the number of levels specified.

4.1.3.3 cspace_err_t cspace_destroy (cspace_t * *c*)

Destroy the specified cspace and return it's resources.

Parameters

c	The specified cspace

Returns

Either CSPACE_ERROR or CSPACE_NOERROR

The routine deletes the caps associated with the level 2 and level 1 cnodes that are associated with this cspace. It also returns the untyped memory back to the untyped memory allocator and frees any book keeping that may have been malloced.

NOTE: The cnode deletion assumes the internal cnode caps have not been copied elsewhere, so that deleting them here results in seL4 having free untyped memory. If the cnode caps have been copied, the memory will not be free in the kernel, but user level will think it is. Thus you will get errors down the track when trying to retype memory that is still in use.

4.1.3.4 seL4_CPtr cspace_alloc_slot (cspace_t * *c*)

Reserve a free slot in the specified cspace.

Parameters

c	Specified cspace

Returns

CSPACE_NULL on error.

This function reserves a slot in the specified cspace that can be used directly with seL4 operations. This function is not normally needed, but can be used for particular optimisations. We assume you know what you are doing in this case. Note: The cspace depth is CSPACE_DEPTH (32) bits for the slots that this function returns (independent of the number of levels).

4.1.3.5 cspace_err_t cspace_free_slot (cspace_t ∗ *c,* seL4_CPtr *slot*)

Return a slot back to the cspace slot allocator.

Parameters

c	The cspace
slot	The slot to free

Returns

Either CSPACE_ERROR or CSPACE_NOERROR

The function puts the slot back into the cspace's free slot list. Note: It assumes there is no capability in the slot (i.e. the slot is actually empty) and it does NOT perform a cspace_delete_cap operation.

This function is mainly used to free the slot used by the cspace_save_reply_cap() routine, as the reply cap deletes itself when used.

4.1.3.6 seL4_CPtr cspace_copy_cap (cspace_t ∗ *dest,* cspace_t ∗ *src,* seL4_CPtr *src_cap,* seL4_CapRights *rights*)

Copy a capability from one cspace to the same or another cspace.

Parameters

dest	The destination cspace.
src	The source cspace.
src_cap	The seL4_CPtr that specifies the capability in source cspace that we copy.
rights	The cap rights can sometime be reduced for some specific caps, but usually this is seL4_All-Rights to create an actual copy.

Returns

seL4_CPtr, CSPACE_NULL on error.

This function implicitly allocates a slot in the destination cspace.

4.1.3.7 cspace_err_t cspace_delete_cap (cspace_t ∗ *c,* seL4_CPtr *cap*)

Delete a specified capability.

Parameters

c	The cspace containing the cap.
cap	The address of the cap to delete.

Returns

Either CSPACE_ERROR or CSPACE_NOERROR

After deleting the capability, the function also implicitly frees the specified slot.

4.1.3.8 seL4_CPtr cspace_mint_cap (cspace_t * *dest*, cspace_t * *src*, seL4_CPtr *src_cap*, seL4_CapRights *rights*, seL4_CapData_t *badge*)

Mint a new capability into the specified cspace.

Parameters

dest	The destination cspace.
src	The source cspace of the capability.
src_cap	The seL4_CPtr of the capability to mint from.
rights	The rights of the new capability, usually seL4_AllRights.
badge	The seL4_CapData of the new capability. It is capability dependent, but either a badge in the case of an EndPoint cap, or a guard for a cnode cap.

Returns

seL4_CPtr, CSPACE_NULL on error.

The function implicitly allocated a slot in the destination cspace.

4.1.3.9 seL4_CPtr cspace_move_cap (cspace_t * *dest*, cspace_t * *src*, seL4_CPtr *src_cap*)

Move a cap from a specified location to another location.

Parameters

dest	The destination cspace (can be the same or different to the source).
src	The source cspace of the capability to move.
src_cap	The seL4_CPtr of the capability to move.

Returns

seL4_CPtr, CSPACE_NULL on error.

Move a capability from one place to another. It implicitly allocates a slot in the destination cspace, and frees the slot in the source cspace.

4.1.3.10 seL4_CPtr cspace_mutate_cap (cspace_t * *dest*, cspace_t * *src*, seL4_CPtr *src_cap*, seL4_CapData_t *badge*)

Move a capability and set it's badge in the process.

Parameters

dest	The destination cspace.
src	The source cspace.
src_cap	The seL4_CPtr to the source capability to move.
badge	The new badge.

Returns

seL4_CPtr, CSPACE_NULL on error.

This function implicitly allocates a slot in the destination cspace and frees the slot on the source.

4.1.3.11 cspace_err_t cspace_recycle_cap (cspace_t * *c*, seL4_CPtr *cap*)

Recycle the specified capability.

Parameters

c	The cspace containing the cap
cap	The seL4_CPtr of the cap to recycle.

Returns

seL4_NOERROR on success.

The function has cap specific behaviour. See the seL4 manual for details. NOTE: There is usually little need for this function (especially if doing COMP9242).

4.1.3.12 cspace_err_t cspace_revoke_cap (cspace_t * *c*, seL4_CPtr *cap*)

Attempt to revoke all capabilities derived from the specified capability.

Parameters

c	The specified cspace.
cap	The seL4_CPtr of the specified capability.

Returns

Either CSPACE_ERROR or CSPACE_NOERROR

This function wraps seL4_CNode_Revoke, see the seL4 manual for a more detailed description. Note: Revoke is usually only needed if you transfer a cap to somebody who can subsequently copy it around. It is used to ensure no copies are still derived from your cap when cleaning up. You should not need it if doing COMP9242.

4.1.3.13 seL4_CPtr cspace_rotate_cap (cspace_t * *dest*, seL4_CapData_t *dest_badge*, cspace_t * *pivot*, seL4_CPtr *pivot_cap*, seL4_CapData_t *pivot_badge*, cspace_t * *src*, seL4_CPtr *src_cap*)

Rotate two caps between three slots.

Don't even think about using this function.

Parameters

dest	
dest_badge	
pivot	
pivot_cap	
pivot_badge	
src	
src_cap	

Returns

4.1.3.14 seL4_CPtr cspace_save_reply_cap (cspace_t * *dest*)

Save the callers reply capability.

Parameters

dest	The destination cspace.

Returns

seL4_CPtr, CSPACE_NULL on error.

This function move the reply capability generated in the process of receiving IPC and moves it from the TCB into the destination cspace. This function implicitly allocates a slot to contain the capability.

NOTE: If the cap is used for a reply, seL4 deletes the capability. cspace_free_slot() should be used to de-allocate the now free slot from this libraries book keeping.

4.1.3.15 seL4_CPtr cspace_irq_control_get_cap (cspace_t ∗ *dest*, seL4_IRQControl *irq_cap*, int *irq*)

Create an IRQ handler capability in the specified cspace.

Parameters

dest	The specified cspace.
irq_cap	The seL4_CPtr of the IRQControl capability in the current cspace.
irq	The irq number of the capability you want to handle.

Returns

CSPACE_NULL on error.

This function implicitly allocates a slot in the specified cspace, and places the new IRQ handler cap into the slot.

4.1.3.16 seL4_Error cspace_ut_retype_addr (seL4_Word *address*, seL4_Word *type*, seL4_Word *size_bits*, cspace_t ∗ *dest*, seL4_CPtr ∗ *dest_cap*)

Create a new kernel object from untyped memory.

Parameters

address	The physical memory address of the location of untyped memory.
type	The type of the seL4 object to create. See libs/libsel4/include/sel4/types.h libs/libsel4/arch_-include/arm/sel4/arch/constants.h appropriate defines.
size_bits	The objects size in bits. What this means precisely is object specific. See the seL4 manual.
dest	The destination cspace to place the capability.
dest_cap	Return the seL4_CPtr location of the new cap to the new object.

Returns

seL4_NOERROR on success.

The function allocates a free slot in the destination cspace to place the capability to the new object. It calls the cspace_ut_translate() (that is provided at bootstrap) to translate the given physical address into a tuple of (untype cap CPtr, offset) to pass to seL4. It then calls to create the object and obtain the capability.

Note: One should cspace_delete_cap() the cap to objects (and all copies of any caps made) in order to return the memory used by the object to free untyped memory.

One could also rely on cspace_destroy() to free object, if, and only if, there are no copies of caps to the object outside of the cspace being destroyed.

4.1.4 Variable Documentation

4.1.4.1 cspace_t∗ cur_cspace

A globally visible for referring to the root tasks current cspace.

It is only valid after the call to cscpace_root_task_bootstrap().

Index