**cspace**

Sel4采用基于capability的访问控制模型，进程对系统资源的访问控制权限组成一个capability space，所有的cap存放在CNode里。CSpace是由CNode构成，CNode可以看成一个数组，数组元素称为slot，每一个slot里面可以存放一个cap或者为空。

CNodes本质是一个结构体数组，该数组的元素是cte（capability table entry），cte中包含一个cap\_t结构体和mdb\_node\_t结构体。cap：包含权限和指向具体cap的指针；

cteMDBNode：保存cap之间的关系，比如：untyped cap衍生出的其它对象的cap。

struct cte {

cap\_t cap;

mdb\_node\_t cteMDBNode;

};

CNodes本身是一个内核对象，也有相应的cap。下面是CNode cap的结构体：

capCNodeGuard：具体的guard数值；

capCNodeGuardSize：guard占的位数；

capCNodeRadix：CNode的最大slot个数；

capCNodePtr：指向CNode的指针；

block cnode\_cap(capCNodeRadix, capCNodeGuardSize, capCNodeGuard, capCNodePtr, capType) {

field capCNodeGuard 64

field capType 5

field capCNodeGuardSize 6

field capCNodeRadix 6

field\_high capCNodePtr 47

}

capability的结构体的大小是16字节，它包含两部分，一部分是seL4\_CapRights，另一部分是指向具体对象cap的指针。

struct cap {

uint64\_t words[2];

};

block seL4\_CapRights {

padding 32

padding 28

field capAllowGrantReply 1

field capAllowGrant 1

field capAllowRead 1

field capAllowWrite 1

}

**Q1 Root Server 构造自己的 Cspace 空间时的步骤：**

1.初始化内存：

Root Server 首先从系统内存中分配一块足够大的区域，用于存储 CNode 和其他需要的数据结构。这个内存区域应当在启动系统时被分配并初始化。

2.创建CNode：

在分配的内存区域中，Root Server 创建一个 CNode 对象。CNode 是一个能力表，其中包含一系列槽（slots），每个槽可以存储一个能力。Root Server 会初始化这个 CNode 并将其作为其 Cspace 的一部分。

3.分配能力：

a.创建能力对象：Root Server 需要为要分配的每个能力创建一个能力对象。这可以是线程、内存区域、设备等资源的能力。

b.初始化能力：对于每个创建的能力对象，Root Server 需要初始化它们的属性，如权限、访问规则等。这些属性将定义谁可以如何访问这些资源。

4.设置默认能力：

Root Server 通常会在其自己的 CNode 中设置默认能力，以便于它能够管理系统资源。这些默认能力通常是对自身的能力，以确保 Root Server 具有足够的权限来初始化和配置系统。

5.传递能力（可选）：

如果 Root Server 需要与其他实体共享资源，它可以将一些能力传递给它们。这涉及在目标实体的 CNode 中创建一个新的能力，然后将原始能力的副本传递给新的能力。这样，目标实体就可以通过自己的 Cspace 访问资源。

6.能力撤销（可选）：

如果需要限制某个实体对资源的访问，Root Server 可以从其 CNode 中删除相关的能力。这将导致该实体失去访问资源的权限。

7.系统初始化完成：

在完成 Cspace 的构造和能力的初始化后，Root Server 可以继续进行其他系统初始化任务，如启动其他线程、设备初始化等。

在 seL4 中，Root Server 的 CNode 中通常包含一些默认构造的能力（capabilities）见下，以便于 Root Server 进行系统的初始化和管理。这些默认能力通常是为了确保 Root Server 具有足够的权限来配置和管理系统资源。

定义了一个 enum cap tag 枚举，其中包含了不同类型的能力（ cap ）以及它们的标签值。  
cap\_null\_cap: 标签值为 0 ，表示一个空的能力。  
cap\_untyped\_cap: 标签值为 2 ，表示未分配的能力，通常用于表示尚未分配给任何特定对象的内存区域。  
cap\_endpoint\_cap: 标签值为 4 ，表示一个端点（ Endpoint ）能力，用于实现进程间通信（ IPC ）。  
cap\_noti昀椀cation\_cap: 标签值为 6 ，表示一个通知（ Noti昀椀cation ）能力，也用于进程间通信。  
cap\_reply\_cap: 标签值为 8 ，表示一个回复（ Reply ）能力，用于在进程间通信中的回复。  
cap\_cnode\_cap: 标签值为 10 ，表示一个 CNode 能力，用于管理能力列表的容器。  
cap\_thread\_cap: 标签值为 12 ，表示一个线程（ Thread ）能力，用于表示一个线程实例。  
cap\_irq\_control\_cap: 标签值为 14 ，表示一个中断控制（ IRQ Control ）能力，用于管理中断。  
cap\_irq\_handler\_cap: 标签值为 16 ，表示一个中断处理（ IRQ Handler ）能力，用于处理中断。  
cap\_zombie\_cap: 标签值为 18 ，表示一个僵尸（ Zombie ）能力，通常用于表示已被撤销的能力。  
cap\_domain\_cap: 标签值为 20 ，表示一个域（ Domain ）能力，用于隔离不同的执行环境。  
cap\_frame\_cap: 标签值为 1 ，表示一个帧（ Frame ）能力，用于内存管理中的页面管理。  
cap\_page\_table\_cap: 标签值为 3 ，表示一个页表（ Page Table ）能力，用于管理虚拟地址与物理地址的映射。  
cap\_asid\_control\_cap: 标签值为 11 ，表示一个 ASID 控制（ ASID Control ）能力，用于管理地址空间标识符。  
cap\_asid\_pool\_cap: 标签值为 13 ，表示一个 ASID 池（ ASID Pool ）能力，用于分配地址空间标识符。

**Q2在 seL4 中，用户态（User Mode）的能力（Capability）通常用于授予用户程序或线程对内核对象的访问权限。**

1.Thread Control：

这个能力允许用户程序控制一个线程，如创建、启动、暂停和终止线程。对应的内核对象是线程（Thread）。

2.IRQ Handler：

这个能力允许用户程序注册并处理中断请求（IRQ）。用户程序可以将这个能力绑定到一个中断处理函数。对应的内核对象是 IRQ。

4.CNode Insert/Remove/Access：

这些能力允许用户程序向 CNode 插入和删除能力，以及访问 CNode 中的能力。对应的内核对象是 CNode。

5.Endpoint Send/Receive：

这些能力允许用户程序发送和接收异步通信的端点。对应的内核对象是 Endpoint。

6.Notification：

这个能力用于异步通知和同步事件。用户程序可以等待通知的触发。对应的内核对象是 Notification。

7.ASID Control：

这个能力允许用户程序控制地址空间标识（ASID），用于虚拟地址到物理地址的映射。对应的内核对象是 ASID 控制。

**Q3在 seL4 中，用户态程序通过能力（Capability）与内核进行通信，访问资源以及执行一些受限的操作。Capability 是一种权限控制的机制，用户态程序只能使用其持有的能力来执行特定的操作。**

1.系统调用：

用户态程序可以通过系统调用接口请求内核执行某些操作，如创建线程、分配内存、发送消息等。在系统调用过程中，用户态程序需要提供相应的参数和能力，以告知内核所需执行的操作。

2.能力传递：

用户态程序可以通过能力传递机制将自己持有的能力传递给其他用户态程序。这允许程序在不同的能力空间中共享资源或协同工作。

3.异步通信：

用户态程序可以使用能力来创建并操作异步通信的对象，如端点（Endpoints）和通知（Notifications）。这允许不同的用户态程序在异步的情况下进行通信和同步。

4.内存管理：

用户态程序可以使用能力来映射和管理虚拟内存和物理内存帧。这允许程序管理其自己的地址空间以及与其他程序共享的内存区域。

**Q4在 seL4 内核态中，能力（Capability）通常用于表示内核对象和资源，以及控制对这些对象的访问权限。不同类型的内核对象对应不同类型的能力，并且每种能力类型都包含一组属性来定义其权限和行为。能力类型以及它们的属性见以下：**

1.Thread Control能力：

- 对应内核对象：线程（Thread）

- 属性：线程的控制权限，如启动、暂停、终止等。

2.IRQ Handler能力：

- 对应内核对象：中断请求（IRQ）

- 属性：用于注册和处理中断请求的能力。

3.CNode Insert/Remove/Access能力：

- 对应内核对象：能力节点（CNode）

- 属性：在 CNode 中插入、删除、访问能力的权限。

4.Endpoint Send/Receive能力：

- 对应内核对象：端点（Endpoint）

- 属性：用于异步通信的发送和接收权限。

5.Notification能力：

- 对应内核对象：通知（Notification）

- 属性：用于异步通知和同步事件的权限。

7.ASID Control能力：

- 对应内核对象：地址空间标识（ASID）

- 属性：对地址空间标识的控制权限。

**一个能力的结构一般包含以下属性**：

-能力标识符（CPtr）：用于在内核中唯一标识能力的整数值。

-能力类型（Type）：指示这个能力的类型，决定了能力的行为和权限。

-权限位（Rights）：描述这个能力的权限，例如读、写、执行等。

-其他相关属性：根据能力类型，可能包括指向内核对象的指针、标志位等。

**Q5在 seL4 中，内核通过一种称为 "capability derivation tree"（能力派生树）的数据结构来维护各种能力的派生关系。能力派生树是一种层次结构，用于表示能力之间的继承和派生关系。每个节点代表一个能力，而树的层次结构表示了能力之间的关系，包括父子关系和派生关系。**

**以下是 seL4 内核维护能力派生关系的基本工作流程**：

1.创建能力：

当一个能力被创建时，内核会为该能力分配一个唯一的能力标识符（CPtr），确定其能力类型和权限等属性。

2.分配给子节点：

当某个能力需要被派生为其他能力时，内核会在派生者（父节点）的派生树下创建一个新的节点（子节点）。父节点的能力将会被子节点继承，包括权限和其他相关属性。

3.能力传递和派生：

当一个能力从一个用户态程序传递到另一个用户态程序时，内核会在目标用户态程序的派生树下创建一个新的节点，继承传递的能力。这样，目标程序就可以使用该能力。

4.Revocation和撤销：

当某个能力被撤销时，内核会删除相应的能力节点，从而取消所有继承该能力的派生能力。这确保了撤销的能力不再能够被使用。

通过这种能力派生树的结构，内核可以轻松地跟踪和维护能力之间的继承和派生关系。这种结构有助于确保能力的正确性、安全性和权限控制，从而防止恶意或错误的使用。能力派生树还可以在不同的用户态程序之间分隔能力，从而实现资源的隔离和访问控制。

**Q6在 seL4 中，用户态通过系统调用与内核交互，管理能力。以下是一些常用的能力管理接口，包括函数签名、作用和参数含义**：

1.seL4\_CNode\_Copy：

- 函数签名：seL4\_CNode\_Copy(seL4\_CPtr dest, seL4\_CPtr dest\_index, seL4\_CPtr src, seL4\_CPtr src\_index, seL4\_Word depth)

- 作用：将一个 CNode 中的能力复制到另一个 CNode。

- 参数：

- dest：目标 CNode 的能力标识符。

- dest\_index：目标 CNode 中的槽索引，表示复制的能力将放置在这个槽中。

- src：源 CNode 的能力标识符。

- src\_index：源 CNode 中的槽索引，表示要复制的能力在源 CNode 中的位置。

- depth：复制深度，用于表示要复制的能力路径的层级。

2.seL4\_CNode\_Move：

- 函数签名：seL4\_CNode\_Move(seL4\_CPtr dest, seL4\_CPtr dest\_index, seL4\_CPtr src, seL4\_CPtr src\_index, seL4\_Word depth)

- 作用：将一个 CNode 中的能力移动到另一个 CNode，同时原 CNode 中的槽将被清空。

- 参数：同上。

3.seL4\_CNode\_Delete：

- 函数签名：seL4\_CNode\_Delete(seL4\_CPtr cnode, seL4\_CPtr index, seL4\_Word depth)

- 作用：从一个 CNode 中删除一个能力。

- 参数：

- cnode：CNode 的能力标识符。

- index：要删除的能力在 CNode 中的槽索引。

- depth：删除深度，表示要删除的能力路径的层级。

这些接口它们通过系统调用提供了用户态与内核之间的能力管理功能。每个接口都具有特定的功能和参数，用于执行不同的操作，如能力复制、能力移动、权限设置等。