FreeRTOS学习笔记（九）

FreeRTOS内存管理

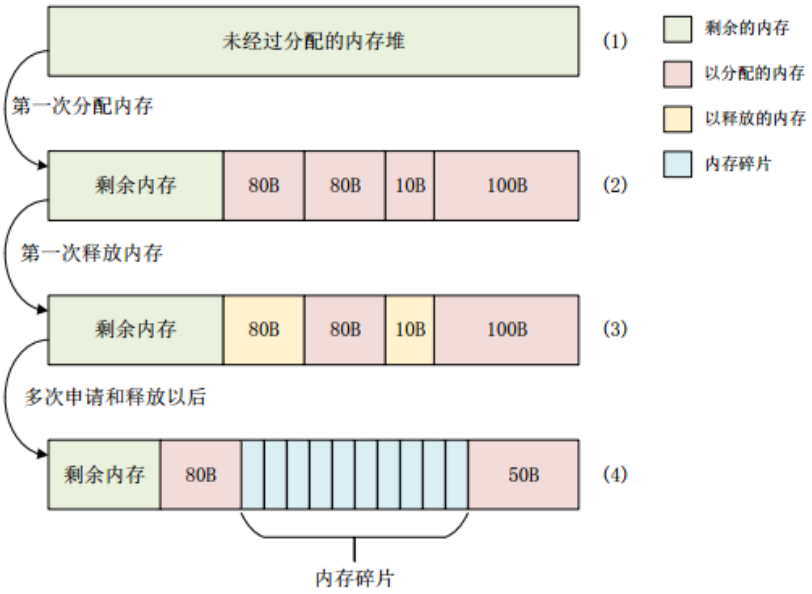

图 17.2.1 内存碎片产生过程

上图可以看出，经过一系列的内存分配之后，内存块越分越小，导致有一堆内存碎片无法被使用。需要用过算法将内存碎片进行合并组成一个新的可用的大内存块。

FreeRTOS内存分配方法

1. heap_1.c
    1. 适用于一些创建好任务，信号量和队列就不会删除的应用
    2. 可以确定执行所花费的时间，不会导致内存碎片
    3. 适用于一些不需要动态内存分配的应用
    4. 可以申请，但是不释放，不合并内存
2. heap_2.c
    1. 可以使用在那些可能重复删除任务、队列、信号量等的应用中
    2. 适合于每次分配和释放大小相同的内存
    3. 如果每次分配内存的大小都不一样，那就要谨慎使用（产生内存碎片）
    4. 可以申请，可以释放，不可以合并
3. heap_3.c
    1. 需要编译器提供一个内存堆，编译器库要提供malloc()和free()函数。比如STM32的话可以通过修改启动文件中的Heap_Size来修改内存堆的带下。
    2. 使用标准C中的malloc()和free()函数分配和释放，并且提供线程保护
4. heap_4.c

1. 可以使用在需要重复创建和删除任务、队列、信号量等的应用中
2. 具有不确定性，比malloc()和free()函数效率高
3. 不会产生严重的内存碎片
4. 可以申请，可以释放，可以合并连续的内存块，不连续的，不可以合并

5. heap_5.c

1. 可以将内部RAM和外部RAM一起作为内存堆使用（heap_4.c只能使用其中一种作为内存堆）
2. 可以申请，可以释放，可以合并连续的内存块，不连续的也可以合并

备注：内存泄漏就是一直申请内存，一值申请内存，导致最后申请不到合适的内存，然后就死机了。

因此，申请malloc()**和释放**free()**是需要成对调用的。（申请了记得要释放）**