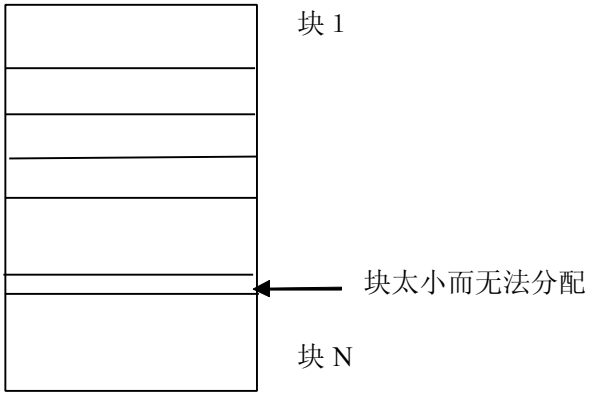
# Ucos 学习之内存管理

## 一、相关背景知识

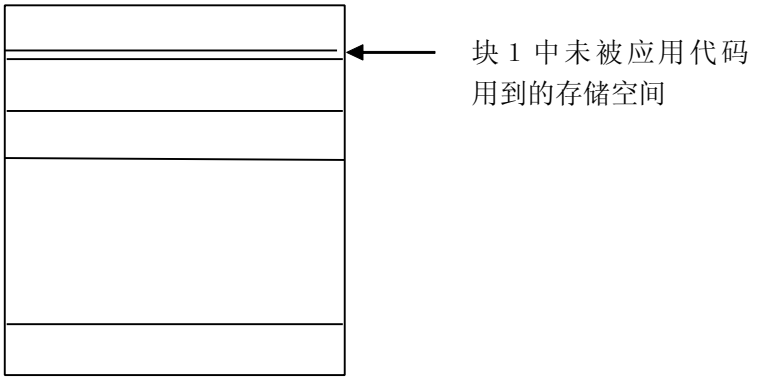
1)、高级语言中的内存管理：在 C ,C++中提供了 malloc(), free(), new, delete 方法用于用户程序的动态存储管理。这些方法，是在称为堆的存储区域中进行内存分配，一般由操作系统、编译器提供支持。基于动态存储分配的内存管理，需要解决由于多次内存分配时存存在的内／外部碎片问题。

外部碎片：指的是在动态存储分配中，多次分配使得在可分配的存储区域中存在一些很难再被分配的小的存储块。




如上图，在多次分配后，堆中存在多个块，其中存在一小块，对于其后的任何一次内存分配请求，该块由于太小而无法被分配。这样的块就称之为内部碎片。内部碎片，一般通过移动存储块，合并小的存储块为较大的存储块的存储紧缩方式进行。使的合并后的小存储块能够再次被利用。在 89x86 的虚拟存储管理中，提供了段式和页式的内存映射，可动态的改变映射表，实现存储地址的重新映射，而不需移动内存块.

内部碎片：一般指在基于固定存储块大小分配的方式中，因为每次只能分配固定大小的存储块，而应用代码实际并不需要申请到的全部空间。有一部分空间已分配但不会被应用代码使用。这也造成浪费。如下图所示：




存储管理，还包括缓存的管理。缓存只存在于 cpu 和外部存储之间的高速存储区域，用于调整的 cpu 处理与慢速的外设之间速度匹配。缓存一般是直接硬件相关的。

## 二、ucos 的存储管理支持

### 1)、分区与块

Ucos 为内存中存储块的动态分配提供了支持。不同于 malloc(), free()，ucos 的存储管理机制是基于固定大小存储块的分配与回收。对于任务而言，每次只能申请固定大小的存储块，因而不存在外部碎片的问题。

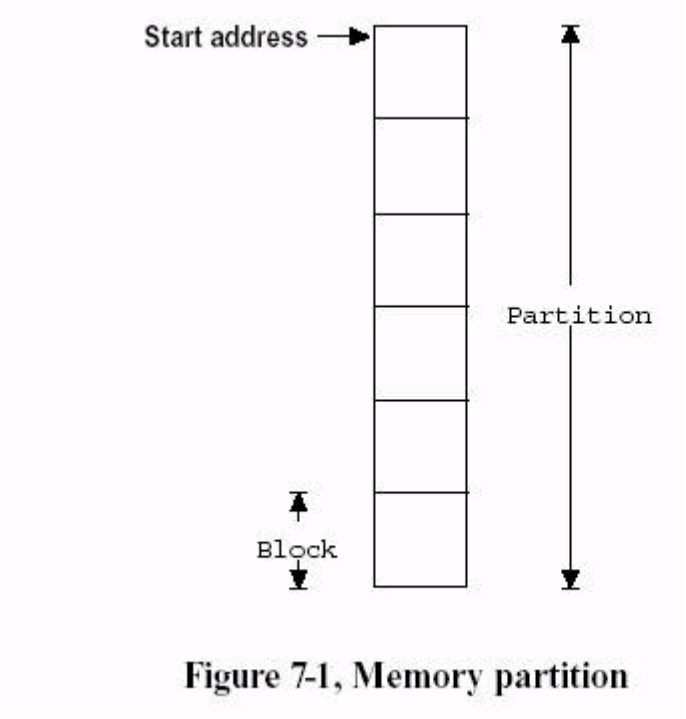



Figure 7-1, Memory partition

如上图所示，ucos 将起始地址为 Start address, 长为 len 的存储区域称为分区( Partition ), 在该分区中划分若干固定大小的块( block )。当任务从该分区请求获取内存空间时，必须以块为单位，不管用户是否需要额外的空间，还是块空间过大。对于块的大小，可由分区创建时设置。因而针对不同的应用，可以创建不同块大小的分区。

### 2)、分区、块相关的数据结构

针对上面提到的分区，由 OS_MEM 提供数据结构支持.

```
typedef struct {
    void     *OSMemAddr;          // 上图中的 Start Address
    void     *OSMemFreeList;      // 用于链表的链接
    INT32U   OSMemBlkSize;        // 上图中 Block 的大小，以字节计
    INT32U   OSMemNBlks;          // 上图分区中总的 Block 数量
    INT32U   OSMemNFree;          // 上图分区中空闲的 Block 数量
} OS_MEM;
```

如结构体各域所示，OS_MEM 描述了一个分区的所有信息。

在 ucos_ii.h 中，有:

```
OS_EXT   OS_MEM          *OSMemFreeList;
OS_EXT   OS_MEM           OSMemTbl[OS_MAX_MEM_PART];
```

OSMemTbl 中各项元素在系统初始化时会构建一单链表。如下图:

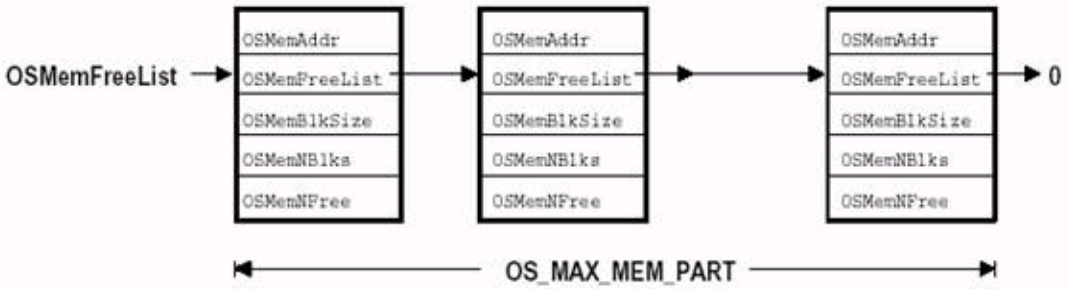
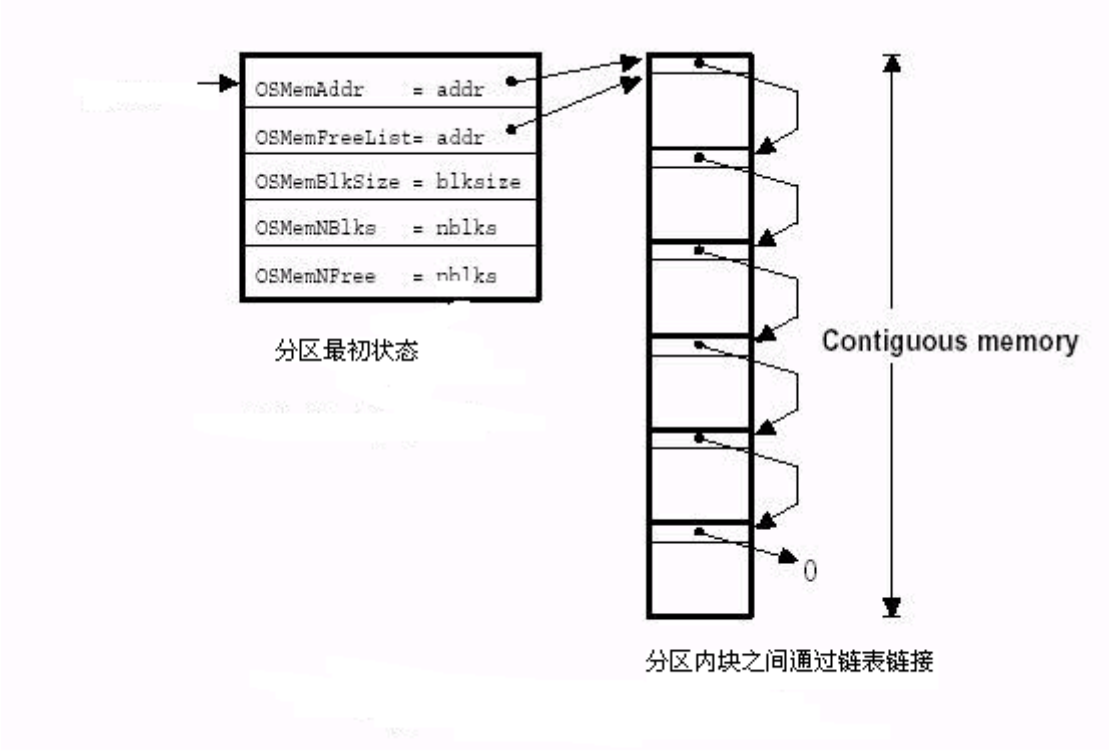



**Figure 7-3, List of free memory control blocks.**

每次在创建分区时，都是从此链表申请一节点，并进行初始化操作.

分区内的各块，则组成一单链表。其中头指针存储在 OS_MEM 域中的.OSMemFreeList 中。每次申请或回收块时，其行为为在块内的链表进行节点的插入与删除操作.如下图所示:




针对单链表的操作，如果有数据结构相关的知识基础，会发现 os_sem 涉及的代码相当容易理解。

**3)、存储操作接口**

主要的操作接口:

```
OS_MEM   *OSMemCreate (void *addr, INT32U nblks, INT32U blksize, INT8U *err); // 分区创建
void   *OSMemGet (OS_MEM *pmem, INT8U *err);      // 从分区获取块
INT8U   OSMemPut (OS_MEM   *pmem, void *pblk)   //   回收块到分区
INT8U   OSMemQuery (OS_MEM *pmem, OS_MEM_DATA *pdata)      // 查询分区状态
void   OS_MemInit (void);          // 分区链表初始化.
```

具体的代码实现，主要涉及链表的操作，以及对 OS_MEM 结构体数据内的信息维护。实现较简单，这里不作说明，可参考相关书籍.

需要注意的是分区内 Block 的链表实现与操作，与 OS_MEM 不同，针对块没有采用单独的数据结构描述，而是直接将下一块的地址存放在块首部，在块没有被分配给任务前，块内的区域无用，这样简化了实现。在块分配给任务后，这个地址无用，所占用的存储空间直接被任务使用。

## 三、应用.

略

## 四、学习分析体会

1、ucos 没有提供复杂的存储管理机制。复杂的机制意味着更多的代码，更多的存储空间和更复杂的操作。可以看到，所涉及的代码主要是链表的操作。

2、存储管理部分没有提供对虚拟存储管理，缓存管理的支持，实际在如 ARM 等 32 位系统中如何应用?