1. 动态存储管理

一．选择题 1C

二．判断题 1．错误 2．正确

三．填空题

1．（1）480+8=488（480 %23+1=0） （2）480-32=448

2．（1）011011110100 （2）011011100000

3．用户不再使用而系统没有回收的结构和变量。例如，p=malloc(size)；…，p=null；

四．应用题

1. 在伙伴系统中，无论占用块或空闲块，其大小均为2的k(k为≥0的正整数)次幂。若内存容量为2m，则空闲块大小只能是20，21，22，…，2m。由同一大块分裂而得的两个小块互称“伙伴空间”，如内存大小为210的块分裂成两个大小为29的块。只有两个“伙伴空间”才能合并成一个大空间。

起始地址为p，大小为2k的内存块，其伙伴的起始地址为：

buddy(p,k)=p+2k  (若p % 2k+1=0),或buddy(p,k)=p-2k  (若p % 2k+1=2k)

2．首次拟合法；从链表头指针开始查找，找到第一个≥所需空间的结点即分配。

最佳拟合法：链表结点大小增序排列，找到第一个≥所需空间的结点即分配。

最差拟合法：链表结点大小逆序排列，总从第一个结点开始分配，将分配后结点所剩空间插入到链表适当位置。

首次拟合法适合事先不知道请求分配和释放信息的情况,分配时需查询,释放时插在表头。 最佳拟合法适用于请求分配内存大小范围较宽的系统,释放时容易产生存储量很小难以利用的内存碎片,同时保留那些很大的内存块以备将来可能发生的大内存量的需求，分配与回收均需查询。 最差拟合法适合请求分配内存大小范围较窄的系统,分配时不查询,回收时查询,以便插入适当位置。

3． 011011110100 4．011011100000

5．（1）buddy(1664,7)=1664-128=1536 （2）buddy(2816,6)=2816+64=2880

6．动态存储分配伙伴系统的基本思想请参见上面题1。边界标识法在每块的首尾均有“占用”/“空闲”标志，空闲块合并方便。伙伴系统算法简单，速度快，但只有互为伙伴的两个空闲块才可合并，因而易产生虽空闲但不能归并的碎片。

7．组织成循环链表的可利用空间表的结点大小按递增序排列时, 首次适配策略就转变为最佳适配策略。

8．因为512=29,可利用空间表的初始状态图如8-1所示。

当用户申请大小为23的内存块时,因24<23<=25,但没有大小为25的块,只有大小为29的块,故将29的块分裂成两个大小为28的块,其中大小为28的一块挂到可利用空间表上,另一块再分裂成两个大小为27的块。又将其中大小为27的一块挂到可利用空间表上,另一块再分裂成两个大小为26的块,一块26的块挂到可利用空间表上,另一块分裂成两个大小为25的块,其中一块挂到可利用空间表上,另一块分给用户(地址0—31)。如此下去,最后每个用户得到的存储空间的起始地址如图8-2, 6个用户分配所需要的存储空间后可利用空间表的状态如图8-3。

在回收时,因为给申请45的用户分配了26,其伙伴地址是0,在占用中,不能合并,只能挂到可利用空间表上。在回收大小为52的占用块时,其伙伴地址是192,也在占用。回收大小为11的占用块时,其伙伴地址是48,可以合并为大小25的块, 挂到可利用空间表上。回收3个占用块之后可利用空间表的状态如图8-4。



|  |  |
| --- | --- |
| 存储大小 | 起始地址 |
| 23 | 0 |
| 45 | 64 |
| 52 | 128 |
| 100 | 256 |
| 11 | 32 |
| 19 | 192 |

图8-2 图8-1

（注：在图8.3和8.4画上了占用块，从原理上，只有空闲块才出现在“可利用空间表”中。）



图8-3 图8-4

9． 因为768 % 27+1=0，所以768和768+27=896互为伙伴, 伙伴合并后,首址为768,块大小为28。因为768 % 28+1=28,所以,所以首址768大小为28的块和首址512大小为28的块合并,成为首址512大小为29的空闲块。因为128 % 27+1=27，其伙伴地址为128-27=0, 将其插入可利用空间表中。回收后该伙伴系统的状态图如下。



10．（1）系统回收一个起始地址为559，大小为45的空闲块后，因右侧起始地址604为空闲块，应与之合并。合并后，起始地址为559，大小为167的空闲块。链表状态如图10．（1）所示。



10．（1）

（2）系统在接受存储块大小为100的请求后，将大小为117的空闲块分出100给予用户。在回收一个起始地址为515，大小为44的空闲块之后，因左侧起始地址462大小53和右侧起始地址559大小167均为空闲块，应与之合并。合并后，起始地址为462，大小为264的空闲块。链表状态如图10．（2）所示。



10．（2）