宏(第 9 行)将大小和已分配位结合起来并返回一个值，可以把它存放在头部或者脚部中。

code/vm/malloc/mm.c

```
1  /* Basic constants and macros */
2  #define WSIZE       4       /* Word and header/footer size (bytes) */
3  #define DSIZE       8       /* Double word size (bytes) */
4  #define CHUNKSIZE  (1<<12)  /* Extend heap by this amount (bytes) */
5
6  #define MAX(x, y) ((x) > (y)? (x) : (y))
7
8  /* Pack a size and allocated bit into a word */
9  #define PACK(size, alloc)  ((size) | (alloc))
10
11 /* Read and write a word at address p */
12 #define GET(p)       (*(unsigned int *)(p))
13 #define PUT(p, val)  (*(unsigned int *)(p) = (val))
14
15 /* Read the size and allocated fields from address p */
16 #define GET_SIZE(p)  (GET(p) & ~0x7)
17 #define GET_ALLOC(p) (GET(p) & 0x1)
18
19 /* Given block ptr bp, compute address of its header and footer */
20 #define HDRP(bp)       ((char *)(bp) - WSIZE)
21 #define FTRP(bp)       ((char *)(bp) + GET_SIZE(HDRP(bp)) - DSIZE)
22
23 /* Given block ptr bp, compute address of next and previous blocks */
24 #define NEXT_BLKP(bp)  ((char *)(bp) + GET_SIZE(((char *)(bp) - WSIZE)))
25 #define PREV_BLKP(bp)  ((char *)(bp) - GET_SIZE(((char *)(bp) - DSIZE)))
```

code/vm/malloc/mm.c

图 9-43　操作空闲链表的基本常数和宏

GET 宏(第 12 行)读取和返回参数 p 引用的字。这里强制类型转换是至关重要的。参数 p 典型地是一个(viod*)指针，不可以直接进行间接引用。类似地，PUT 宏(第 13 行)将 val 存放在参数 p 指向的字中。

GET_SIZE 和 GET_ALLOC 宏(第 16～17 行)从地址 p 处的头部或者脚部分别返回大小和已分配位。剩下的宏是对块指针(block pointer，用 bp 表示)的操作，块指针指向第一个有效载荷字节。给定一个块指针 bp，HDRP 和 FTRP 宏(第 20～21 行)分别返回指向这个块的头部和脚部的指针。NEXT_BLKP 和 PREV_BLKP 宏(第 24～25 行)分别返回指向后面的块和前面的块的块指针。

可以用多种方式来编辑宏，以操作空闲链表。比如，给定一个指向当前块的指针 bp，我们可以使用下面的代码行来确定内存中后面的块的大小：

```
size_t size = GET_SIZE(HDRP(NEXT_BLKP(bp)));
```

3. 创建初始空闲链表

在调用 mm_malloc 或者 mm_free 之前，应用必须通过调用 mm_init 函数来初始化堆(见图 9-44)。

mm_init 函数从内存系统得到 4 个字，并将它们初始化，创建一个空的空闲链表(第 4～10 行)。然后它调用 extend_heap 函数(图 9-45)，这个函数将堆扩展 CHUNKSIZE 字