code/vm/malloc/memlib.c

```
/* Private global variables */
static char *mem_heap;     /* Points to first byte of heap */
static char *mem_brk;      /* Points to last byte of heap plus 1 */
static char *mem_max_addr; /* Max legal heap addr plus 1*/

/*
 * mem_init - Initialize the memory system model
 */
void mem_init(void)
{
    mem_heap = (char *)Malloc(MAX_HEAP);
    mem_brk = (char *)mem_heap;
    mem_max_addr = (char *)(mem_heap + MAX_HEAP);
}

/*
 * mem_sbrk - Simple model of the sbrk function. Extends the heap
 *    by incr bytes and returns the start address of the new area. In
 *    this model, the heap cannot be shrunk.
 */
void *mem_sbrk(int incr)
{
    char *old_brk = mem_brk;

    if ( (incr < 0) || ((mem_brk + incr) > mem_max_addr)) {
        errno = ENOMEM;
        fprintf(stderr, "ERROR: mem_sbrk failed. Ran out of memory...\n");
        return (void *)-1;
    }
    mem_brk += incr;
    return (void *)old_brk;
}
```

code/vm/malloc/memlib.c

图 9-41 memlib.c：内存系统模型

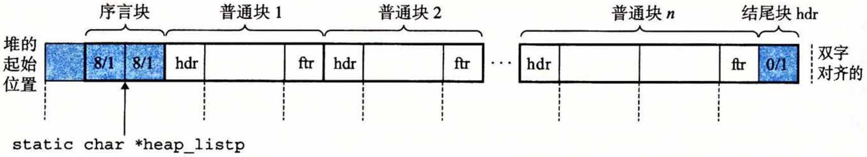

图 9-42 隐式空闲链表的恒定形式

2. 操作空闲链表的基本常数和宏

图 9-43 展示了一些我们在分配器编码中将要使用的基本常数和宏。第 2～4 行定义了一些基本的大小常数：字的大小(WSIZE)和双字的大小(DSIZE)，初始空闲块的大小和扩展堆时的默认大小(CHUNKSIZE)。

在空闲链表中操作头部和脚部可能是很麻烦的，因为它要求大量使用强制类型转换和指针运算。因此，我们发现定义一小组宏来访问和遍历空闲链表是很有帮助的(第 9～25 行)。PACK