

15213 Recitation Section C

Shimin Chen
Nov. 11, 2002

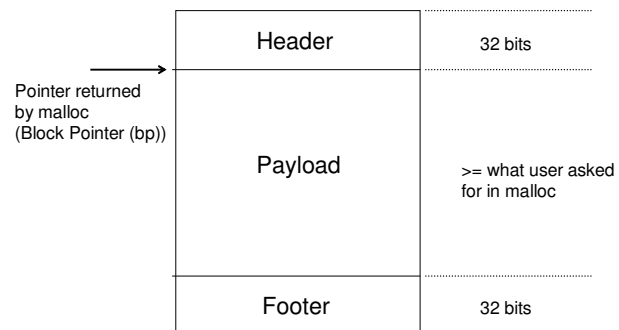
Outline

- Understanding mm-helper.c
- Adding debugging info to mm-helper.c

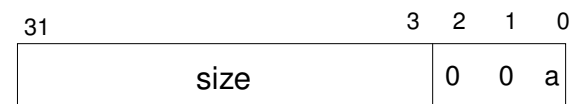
What does mm-helper.c do ?

- Implicit Free List
 - Header with each block – (size / allocated bit)
 - No separate Free List – free blocks linked implicitly by size fields in header
- First Fit
 - Searches free list from beginning and picks first block that is large enough
- Immediate Boundary Tag Coalescing
 - Footer (boundary tag), replica of header

Block Format

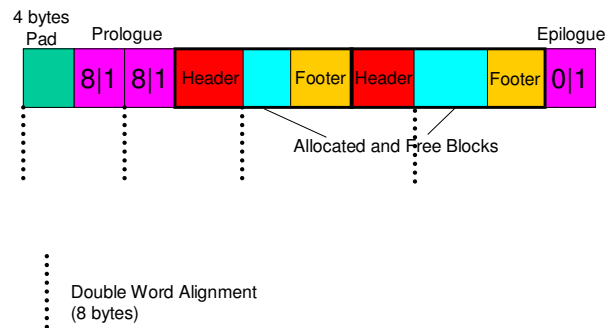


Header/Footer Format



- Double word alignment
 - Three lower-order bits of size always 0
- Pack size and allocated bits into a single integer
 - Size = 24 (0x18). Block is allocated
 - Header = 0x18 | 0x1 = 0x19

Heap Format



Very Useful Macros

- `#define WSIZE 4`
- `#define DSIZE 8`
- `#define CHUNKSIZE (1<<12)`
- `#define OVERHEAD 8`

Very Useful Macros

- `#define PACK(size, alloc) ((size) | (alloc))`
- `#define GET(p) (*(size_t*)(p))`
- `#define PUT(p, val) (*(size_t*)(p) = (val))`
- `#define GET_SIZE(p) (GET(p) & ~0x7)`
- `#define GET_ALLOC(p) (GET(p) & 0x1)`

Very Useful Macros

- `#define HDRP(bp)`
`((char*)(bp) - WSIZE)`
- `#define FTRP(bp)`
`((char*)(bp) + GET_SIZE(HDRP(bp)) - DSIZE)`
- `#define NEXT_BLKP(bp)`
`((char*)(bp) + GET_SIZE(((char*)(bp) - WSIZE)))`
- `#define PREV_BLKP(bp)`
`((char*)(bp) - GET_SIZE(((char*)(bp) - DSIZE)))`

Initializing the Heap

```
int mm_init(void) {
    if ((heap_listp = mem_sbrk(4*WSIZE)) == NULL)
        return -1;

    PUT(heap_listp, 0);
    PUT(heap_listp+WSIZE, PACK(OVERHEAD, 1));
    PUT(heap_listp+DSIZE, PACK(OVERHEAD, 1));
    PUT(heap_listp+WSIZE+DSIZE, PACK(0, 1));
    heap_listp += DSIZE;

    if (extend_heap(CHUNKSIZE/WSIZE) == NULL)
        return -1;

    return 0;
}
```

15213 Recitation C

9

Shimin Chen

Extending the Heap

```
static void *extend_heap(size_t words) {
    char *bp;
    size_t size;

    size = (words % 2) ? (words+1)*WSIZE :
              words*WSIZE;
    if ((int)(bp = mem_sbrk(size)) < 0)
        return NULL;

    PUT(HDRP(bp), PACK(size, 0));
    PUT(FTRP(bp), PACK(size, 0));
    PUT(HDRP(NEXT_BLKP(bp)), PACK(0, 1));

    return coalesce(bp);
}
```

15213 Recitation C

10

Shimin Chen

Malloc

```
void *mm_malloc(size_t size) {
    size_t asize, extendsize;
    char *bp;

    if (size <= 0) return NULL;
    if (size <= DSIZE)
        asize = DSIZE+OVERHEAD;
    else
        asize = DSIZE*((size+(OVERHEAD)+(DSIZE-1))/DSIZE);

    if ((bp = find_fit(asize)) != NULL) { ←
        place(bp, asize);
        return bp;
    }
    extendsize = MAX(asize, CHUNKSIZE);
    if ((bp = extend_heap(extendsize/WSIZE)) == NULL)
        return NULL;
    place(bp, asize);
    return bp;
}
```

15213 Recitation C

11

Shimin Chen

Finding First Fit

```
static void *find_fit(size_t asize) {
    void *bp;

    for (bp = heap_listp; GET_SIZE(HDRP(bp)) > 0;
         bp = NEXT_BLKP(bp))
        if (!GET_ALLOC(HDRP(bp))
            && (asize <= GET_SIZE(HDRP(bp))))
            return bp;

    return NULL;
}
```

15213 Recitation C


12

Shimin Chen

Malloc

```
void *mm_malloc(size_t size) {
    size_t asize, extendsize;
    char *bp;

    if (size <= 0) return NULL;
    if (size <= DSIZE)
        asize = DSIZE+OVERHEAD;
    else
        asize = DSIZE*((size+(OVERHEAD)+(DSIZE-1))/DSIZE);

    if ((bp = find_fit(asize)) != NULL) {
        place(bp, asize); 
        return bp;
    }
    extendsize = MAX(asize, CHUNKSIZE);
    if ((bp = extend_heap(extendsize/WSIZE)) == NULL)
        return NULL;
    place(bp, asize);
    return bp;
}
```

15213 Recitation C

13

Shimin Chen

Placing a Block in a Free Block

```
static void place(void *bp, size_t asize) {
    size_t csize = GET_SIZE(HDRP(bp));

    if ((csize - asize) >= (DSIZE + OVERHEAD)) {
        PUT(HDRP(bp), PACK(asize, 1));
        PUT(FTRP(bp), PACK(asize, 1));
        bp = NEXT_BLKP(bp);
        PUT(HDRP(bp), PACK(csize-asize, 0));
        PUT(FTRP(bp), PACK(csize-asize, 0));
    }
    else {
        PUT(HDRP(bp), PACK(csize, 1));
        PUT(FTRP(bp), PACK(csize, 1));
    }
}
```

15213 Recitation C

14

Shimin Chen

Malloc

```
void *mm_malloc(size_t size) {
    size_t asize, extendsize;
    char *bp;

    if (size <= 0) return NULL;
    if (size <= DSIZE)
        asize = DSIZE+OVERHEAD;
    else
        asize = DSIZE*((size+(OVERHEAD)+(DSIZE-1))/DSIZE);

    if ((bp = find_fit(asize)) != NULL) {
        place(bp, asize);
        return bp;
    }
    extendsize = MAX(asize, CHUNKSIZE);
    if ((bp = extend_heap(extendsize/WSIZE)) == NULL)
        return NULL;
    place(bp, asize);
    return bp;
}
```

15213 Recitation C

15

Shimin Chen

Free

```
void mm_free(void *bp) {
    size_t size = GET_SIZE(HDRP(bp));

    PUT(HDRP(bp), PACK(size, 0));
    PUT(FTRP(bp), PACK(size, 0));

    coalesce(bp);
}
```

15213 Recitation C

16

Shimin Chen

Coalesce: Called by mm_free() & extend_heap()

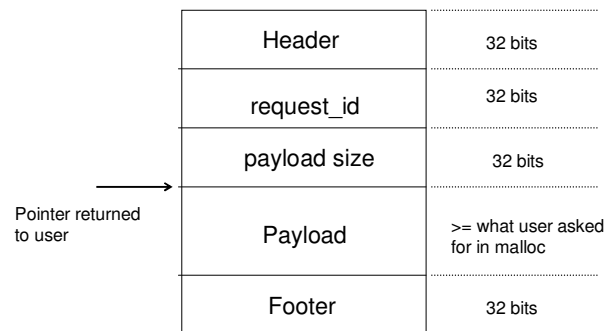
```
static void *coalesce(void *bp) {
    size_t prev_alloc = GET_ALLOC(FTRP(PREV_BLKP(bp)));
    size_t next_alloc = GET_ALLOC(HDRP(NEXT_BLKP(bp)));
    size_t size = GET_SIZE(HDRP(bp));

    if (prev_alloc && next_alloc) { return bp; }
    else if (prev_alloc && !next_alloc) { ..... }
    else if (!prev_alloc && next_alloc) {
        size += GET_SIZE(HDRP(PREV_BLKP(bp)));
        PUT(FTRP(bp), PACK(size, 0));
        PUT(HDRP(PREV_BLKP(bp)), PACK(size, 0));
        bp = PREV_BLKP(bp);
    }
    else { ..... }
    return bp;
}
```

Adding Debugging Information

- mm_heapcheck
- Display “request_id” and “payload” of every active block
 - request_id : malloc request counter (0..
 - mm_init sets the counter to 0
 - mm_malloc increments the counter
 - payload size : the memory requested by malloc
 - Can be different from the allocated size!!!
- We need to store the info somewhere ?

Allocated Block Format



One Way to Implement This

- Inside malloc
 - Allocate additional memory in malloc

```
PUT(bp,request_counter);
PUT(bp+4,size);
return bp+DSIZE;
```

- Inside Free
 - bp = bp - DSIZ;

Heapcheck

- Put all sorts of sanity checks
- Scan the implicit list
 - like the first fit function
 - print request_id and size

Explicit Lists

- Separate Free List
 - Can find a free block quickly
- Change Free Block Format
 - Add prev pointer
 - Add next pointer
- Where to store free list pointer
 - Only one WORD
 - Can store in unused PAD word
- Some functions to add
 - `static void insertfree_block(void * freeblkptr);`
 - `static void removefree_block(void * freeblkptr);`