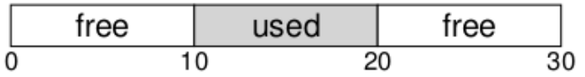
OSTEP Chapter 17

ECE 3600, Fall 2022

Table of Contents

1. Free-Space Management

fragmentation, splitting, coalescing



The free list for this heap would have two elements on it. One entry describes the first 10-byte free segment (bytes 0-9), and one entry describes the other free segment (bytes 20-29):

After `malloc(1)`:

vs. after `free(10)`:

After merge:

2. Implementation

```
ptr = malloc(20);
```

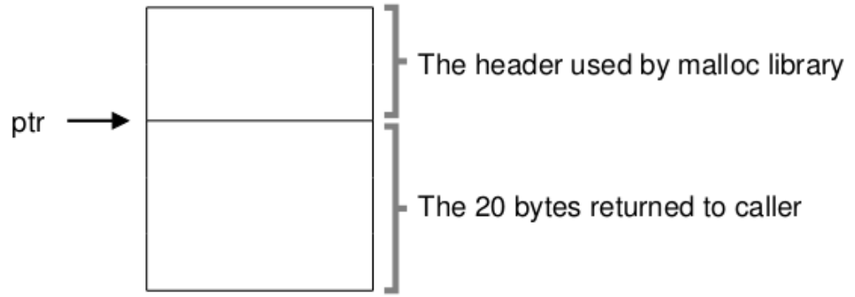


Figure 17.1: **An Allocated Region Plus Header**

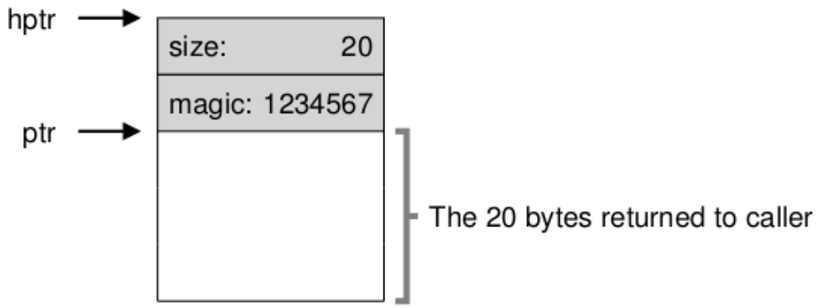


Figure 17.2: **Specific Contents Of The Header**

```
typedef struct { int size; int magic; } header_t;

void free(void *ptr) { header_t *hptr = (header_t *) ptr - 1; ... }
```

3. Heap with One Allocation

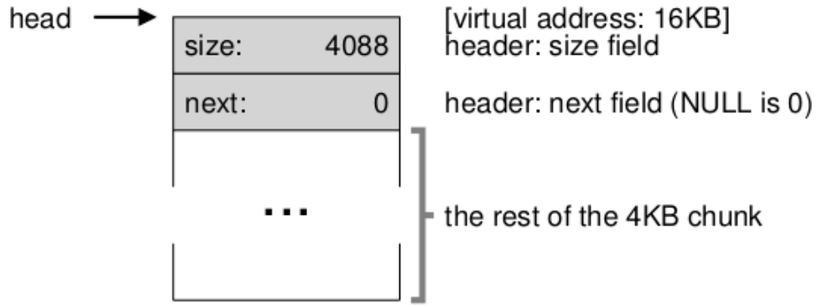


Figure 17.3: **A Heap With One Free Chunk**

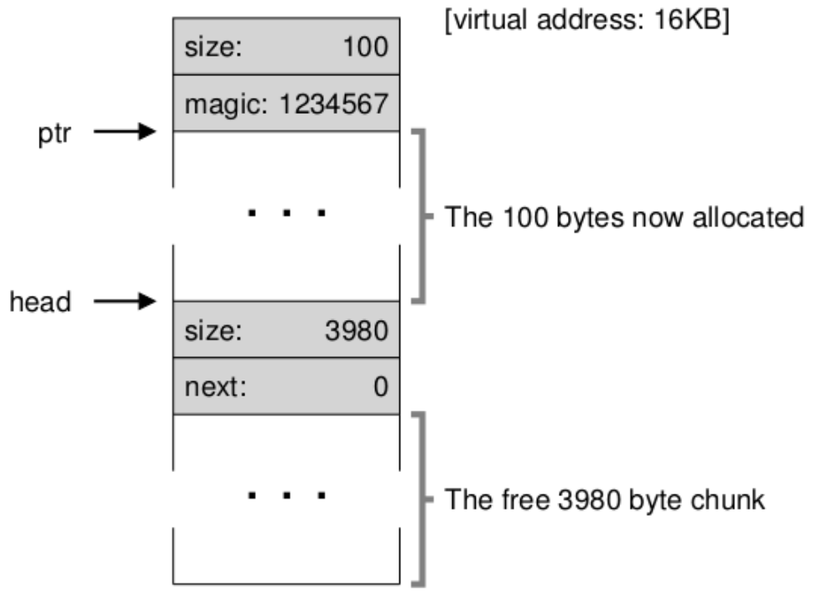


Figure 17.4: **A Heap: After One Allocation**

```
4096 - 108 - 8 = 3980
```

4. Heap with Three Allocations

```
16 KB = 16384;  sptr = 16384 + 108 + 8 = 16500;  head = sptr + 100 + 108 = 16708
```

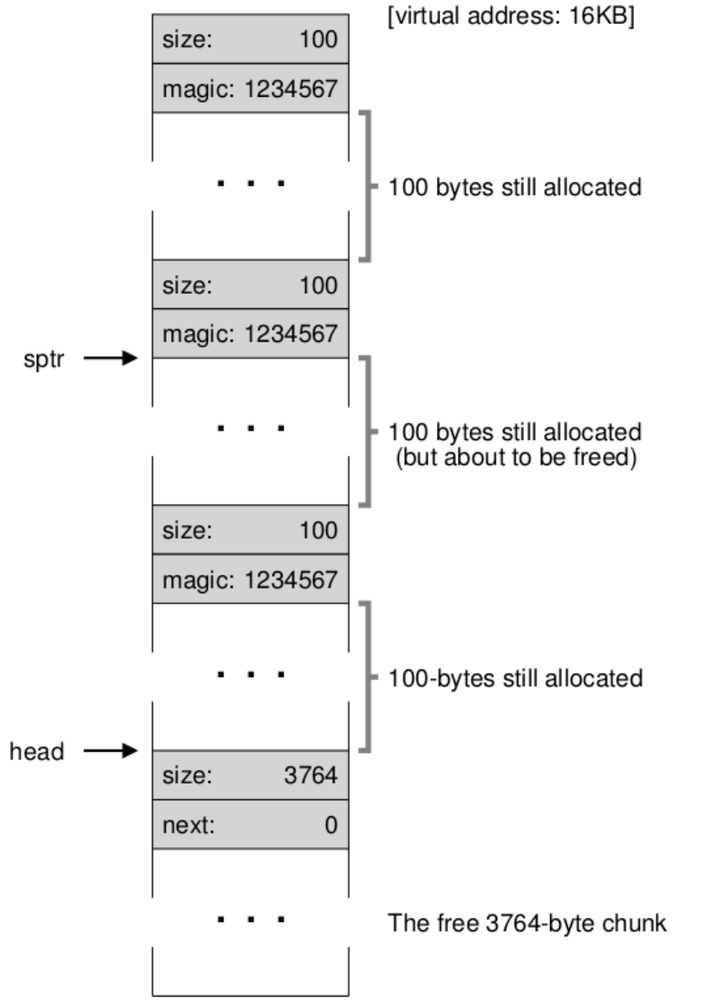


Figure 17.5: **Free Space With Three Chunks Allocated**

```
4096 - 3*108 - 8 = 3764
```

5. Heap with Two Allocations

```
after free(sptr): head = sptr - 8 = 16492;
```

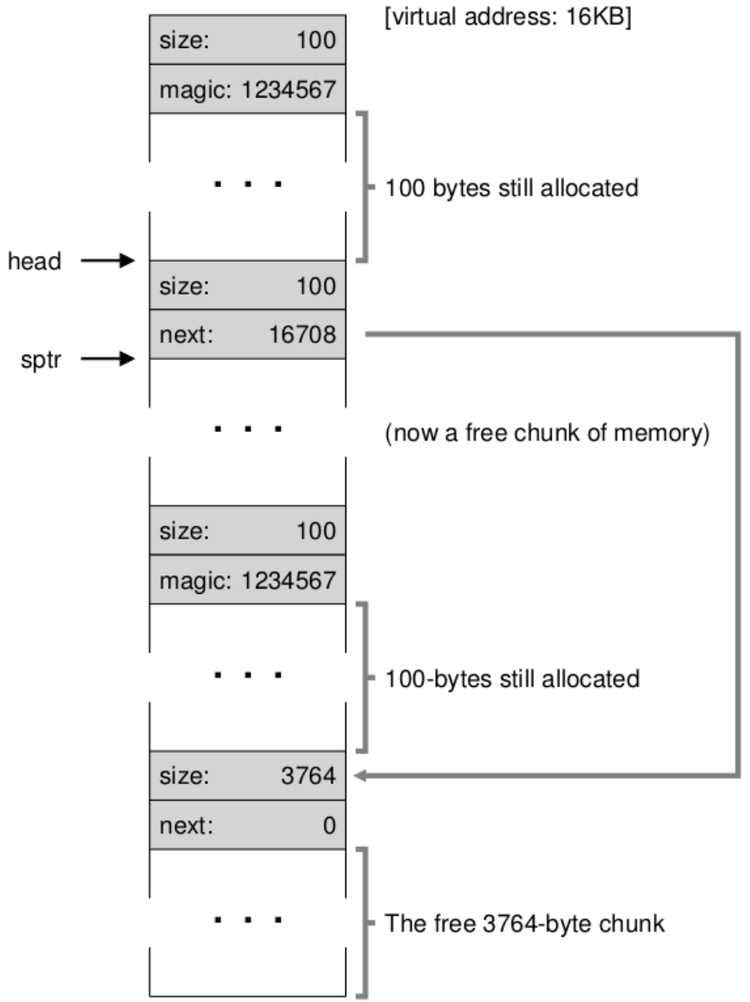


Figure 17.6: **Free Space With Two Chunks Allocated**

6. A Non-Coalesced Free List

after last two in-use chunks freed, without coalescing:

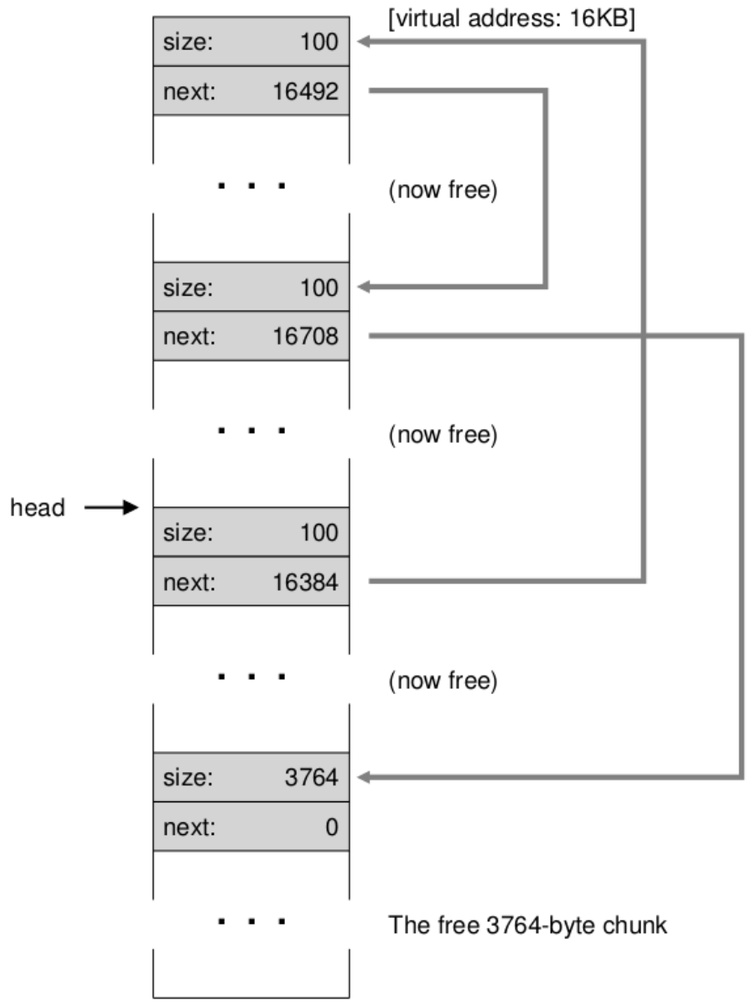


Figure 17.7: **A Non-Coalesced Free List**

7. Policies

goals: minimize fragmentation, minimize search time

ordering: address-based vs. increasing size vs. decreasing size

Best Fit, Worst Fit, First Fit, Next Fit

Assume an allocation request of size 15. A best-fit approach would search the entire list and find that 20 was the best fit, as it is the smallest free space that can accommodate the request. The resulting free list:

As happens in this example, and often happens with a best-fit approach, a small free chunk is now left over. A worst-fit approach is similar but instead finds the largest chunk, in this example 30. The resulting list:

The first-fit strategy, in this example, does the same thing as worst-fit, also finding the first free block that can satisfy the request. The difference is in the search cost; both best-fit and worst-fit look through the entire list; first-fit only examines free chunks until it finds one that fits, thus reducing search cost.

Other: tree vs. list, separate lists for different sizes, custom block allocation, ...

8. Custom Block Allocation

An application with heavy use of malloc() and free() for a particular data structure may obtain a significant performance increase using block allocation and caching.

For example, data nodes could be allocated in blocks of 1000, so then to allocate 1,000,000 nodes malloc() would be called only 1000 times.

And instead of using free(), unneeded nodes could be saved for later reuse.

Example implementation:

```
void *emalloc( size_t nbytes) { // malloc() with exit on error
  void *p = malloc(nbytes); if(!p) { fprintf( stderr, "malloc() failed\n"); exit(1); } return p;
}

typedef struct node { double x; int y; /* ... data ... */ struct node *next; } Node;

#if BLOCK
static Node *head = 0;
static int block = 1000; // number of nodes to allocate at one time
Node *new_node( void) {
  if( !head) { // allocate a block of nodes
    head = emalloc(block*sizeof(Node)); Node *last = head+block-1;
    for( Node *p = head; p < last; ++p) p->next = p+1; // link the nodes
    last->next = 0;
  }
  Node *n = head; head = head->next; return n; // return head of list
}
void free_node( Node *n) {
  n->next = head; head = n; // insert at head of list
}
#else
#define new_node()   emalloc(sizeof(Node))
#define free_node(n) free(n)
#endif
```

9. Exercises

Exercises from the book using malloc.py:

1. First run with the flags -n 10 -H 0 -p BEST -s 0 to generate a few random allocations and frees. Can you predict what alloc()/free() will return? Can you guess the state of the free list after each request? What do you notice about the free list over time?

2. How are the results different when using a WORST fit policy to search the free list (-p WORST)? What changes?

3. What about when using FIRST fit (-p FIRST)? What speeds up when you use first fit?