CS241 System Programming #11

Memory III: Memory Pools, alignment, Advanced techniques

typedef struct \_pool\_t {

size\_t capacity;

size\_t used;

char buff[]; // C99. Must be last

} pool\_t;

pool\_t\* create\_pool(size\_t capacity) {

// fix the mistake(s)

pool\_t \* result = malloc(capacity + \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_);

assert(result);

result -> capacity = capacity;

memset( result \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, 0x5a, capacity);

return result;

}

void\* allocate(pool\_t\* pool, size\_t request) {

assert(pool);

// How would you round to ensure request is a multiple of sizeof(size\_t) ?

request \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// Leave space for our meta data...

char \* result = pool->buff + used + sizeof(size\_t) \* 2;

// Todo: Round up to ensure natural alignment e.g. result%16 is 0.

result \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

pool->used = request + (result - pool->buff); // Is this correct?

assert( \*result == \_\_\_\_\_ );

size\_t\* bounds = (size\_t\*) result;

bounds[-1] = 0xdeadbeef; // Why this ordering?

bounds[-2] = request;?

bounds[ request ] = 0xBAADF00D; // Fix the error

return result;

}

void free\_all(pool\_t\* pool) { \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ; }

void deallocate(pool\_t\* pool, void\* ptr) {

assert(pool && ptr);

size\_t \*bounds = ptr;

assert(bounds[-1] == \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_);

size\_t size = bounds[-1];

assert( \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_);

memset(ptr, 0x5a, size);

}

Advanced techniques

0. Advantages of memory pools?

1. Using Knuth's *Boundary Tags* to implement coalescing

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 32 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | 32 | 24 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ | 24 | 96 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | 96 |

2. Additional explicit linked list AKA "Segregated free list": Store memory addresses of next free link

- Store free blocks pointers inside the unused space of the free block. More work to do during free()

- Free Block list can now be in arbitrary searchable order (better performance).

3. Segregated free list: Different lists for different sizes. Advantage?

4. Where would you find a SLAB allocator?

5. Advantages of deferred coalescing?

6. Buddy Allocator (example of segregated free list allocator) & Internal Fragmentation

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 16384 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_