Managing the Heap

Sunday, November 8, 2020 5:50 PM

Heap Allocators

Defn. Set of functions for handling heap memory requests.

API.
```
void *malloc(size_t size);
void free(void *ptr);
void *realloc(void *ptr, size_t size);
```

Requirements

→ Arbitrary malloc/free sequences
→ Keep track of allocated, available memory
→ Decide which memory to use to fulfill alloc request
→ Immediately respond to requests without delay

Goals

→ Maximize throughput (minimize time to complete requests)
→ Maximize memory usefulness (minimize fragmentation)

Bump Allocator

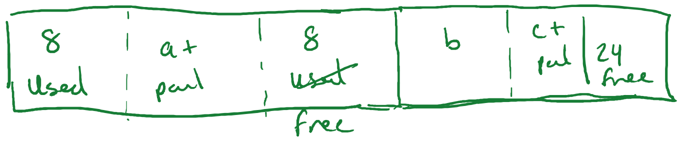
→ Prioritize throughput, don't care about utilization
→ Each malloc only finds next mem location to use
→ Free does nothing
→ Never reuses memory

Implicit Free List Allocator

Motivation. Need to note which blocks are allocated or free

Design. Extra space before each block as header
Stores size of alloc, whether free or not

padding to get to 8n bytes per alloc

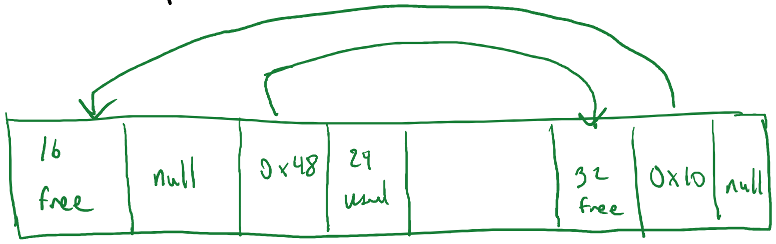


Choosing free block
→ First fit: first free block that fits
→ Next fit: Use next one from previous search
→ Best fit: Search whole heap, use smallest free block

Explicit Free List Allocator

Motivation. Use first 16 bytes of each free blocks to store ptrs to prev, next free blocks

Design



Must enforce minimum 16-byte size of allocs

Now - can jump between free blocks

Can organize LL in address-order or LIFO

Coalescing

Defn. Combine adjacent free blocks

Coalesce when block freed, w/ immediate right neighbor

In-Place Realloc

→ Case 1. Grow, but extra space in padding
- → Just return same address

→ Case 2: Shrink
- → Recycle/split freed block into new block
- → Return same address

→ Case 3: Grow outside padding
- → Try to coalesce w/ immediate right; return same address
- → Otherwise, move data