CS 354 - Machine Organization & Programming
Thursday, October 10, 2019

Project p3 (6%): Assigned Tomorrow, DUE at 10 pm on Monday, October 28th
Homework hw3 (1.5%): Assigned Tomorrow, DUE at 10 pm on Friday, October 18th

Today is last chance to pick up exams from me at lecture.

Last Time

Simple View of Heap
Free Block Organization
Implicit Free List
Placement Policies

Today

Placement Policies (from last time)
Free Block - Too Large/Too Small
Coalescing Free Blocks
Footers
Explicit Free List

Next Time

Finish The Heap
The Memory Hierarchy
Read: B&O 6 intro, 6.2, 6.3

Free Block - Too Large/Too Small

What happens if free block chosen is bigger than the request?

◆ use entire block

– mem util: more internal fragmentation

+ thruput: fast, simple code

- must predivide heap into various-size blocks

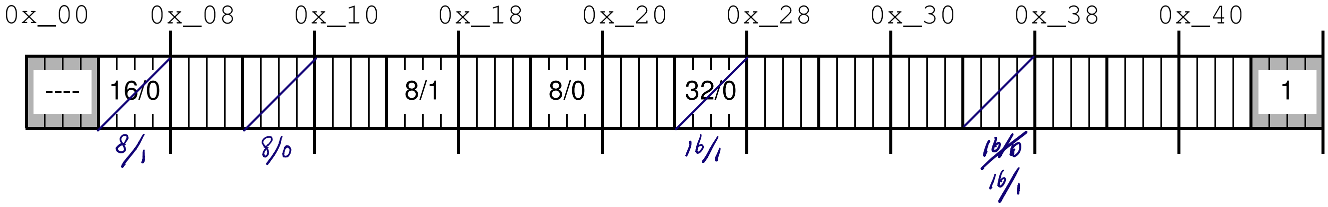
◆ split block, First part is allocated, remaining free

+ mem util: less internal fragmentation.

- thruput: must splict and heap search can become slower as it splinters into smaller pieces.

+ splicting can be done in O(1)

Heap Allocation Run 4 using Splitting and using FF



→ Diagram how the heap above is modified by the 4 mallocs below.
For each, what address is assigned to the pointer?
If there is a new free block, what is its address and size?

	addr:	Free block	size
1) `p1 = malloc(sizeof(char));` 8	0x_08	0x_0C	8
2) `p2 = malloc(11 * sizeof(char));` 16	0x_28	0x_38	16
3) `p3 = malloc(2 * sizeof(int));` 16	0x_38	no new free block	
4) `p4 = malloc(5 * sizeof(int));` 24	0x_00	Alloc Fails	

What happens if there isn't a large enough free block to satisfy the request?

1st. coalesce free blocks

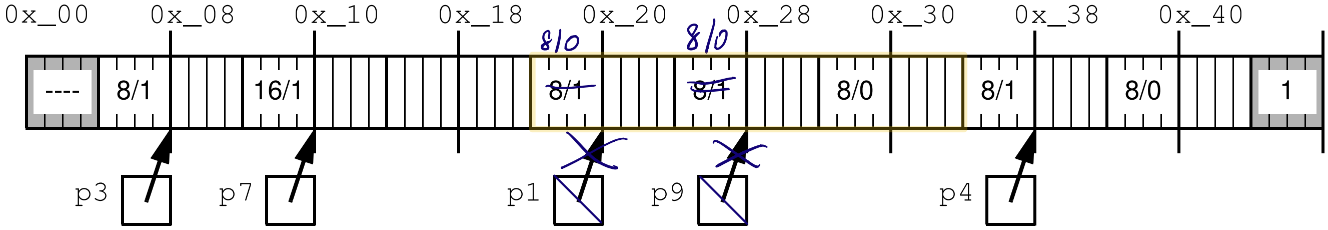
→ Can allocated blocks be moved out of the way to create larger free areas? [✓ No]

2nd. Ask kernel for more heap.

3rd. Return null indicating request can't be satisfied

Coalescing Free Blocks

Heap Allocation Run 5 without Coalescing



→ What's the problem resulting from the following heap operations using FF?

```
1) free(p9); p9 = NULL;
2) free(p1); p1 = NULL;
3) p1 = malloc(4 * sizeof(int));
```

ALLOC Fail.

Problem: false fragmentation (External)

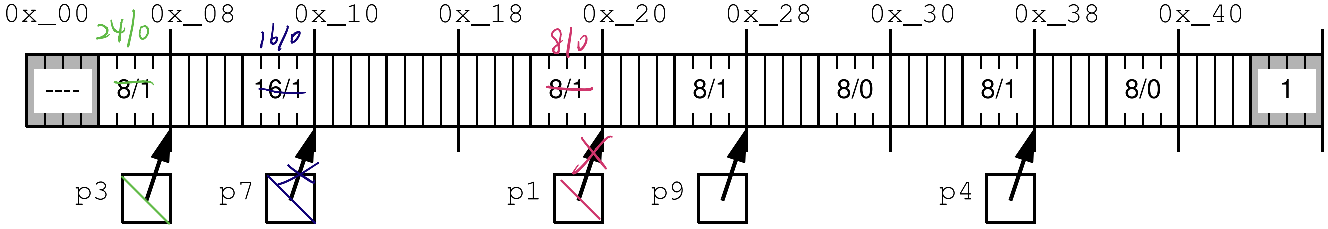
is when heap request can not be satisfied despite their being large enough area of heap - but is divided into smaller blocks.

Solution: Coalesce adjacent free blocks.

* <u>immediate</u>: coalesce every time block is freed

<u>delayed</u>: only coalesce when needed to satisfied a request for larger block.

Heap Allocation Run 6 with Immediate Coalescing



→ Given the heap above, what is the size in bytes of the freed heap block?

```
1) free(p7); p7 = NULL;
```

→ Given a pointer to a payload, how do you find its block header?

pointer - 4 bytes

→ Given a pointer to a payload, how do you find the block header of the next block?

pointer - 4 bytes + block_size bytes

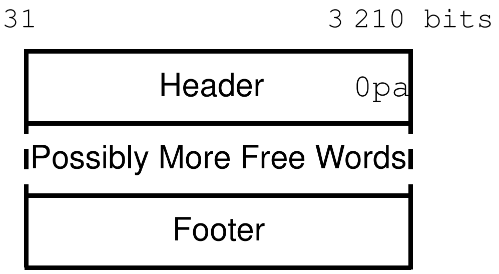
→ Given the modified heap above, what is the size in bytes of the freed heap block when immediate coalescing is used?

```
2) free(p3); p3 = NULL;
3) free(p1); p1 = NULL;
```

➢ Given a pointer to a payload, how do you find the block header of the previous block?

Footers

Heap FREE Block Layout with Header and Footer



Footer (AKA Boundary Tag)
Last word in a free block containing just size

p bit status of Previous block
0 = prev ~~blo~~block free
1 = prev block allocated.

→ Why don't allocated blocks need footers?
Allocated blocks are not coalesce

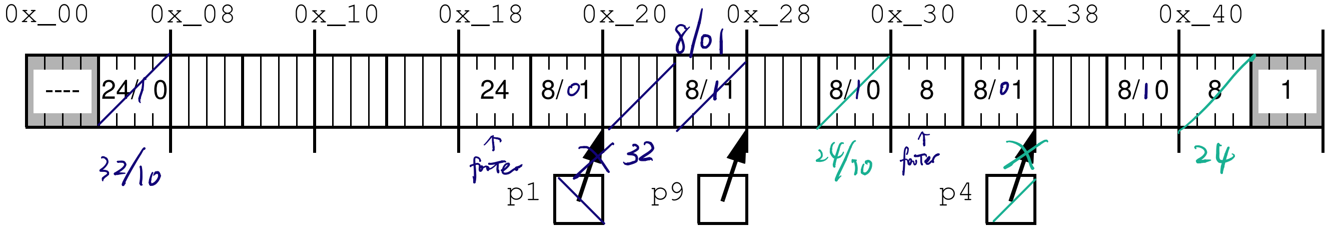
→ If only free blocks have footers, how do we know if previous block will have a footer?
check current blocks' p Bit.

→ What integer value will the header have for an allocated block that is

1) 8 bytes in size and prev. block is free? 8+0+1 = 9 8/01
2) 8 bytes in size and prev. block is allocated? 8+2+1 = 11 8/11
3) 32 bytes in size and prev. block is allocated? 32+2+1 = 35 32/3
4) 64 bytes in size and prev. block is free?? 64+2+1 = 67 64/3

→ Given a pointer to a payload, how do you get to the header of a previous block if it's free?
1. pointer − 4 bytes set to curr block header
2. if p Bit = Ø (previous) Then pointer − 8 gets to footer with size.
3. pointer − 4 bytes − prev-block-size bytes to get to prev block header

Heap Allocation Run 7 with Immediate Coalescing and Free Block Footers



→ Given the heap above, what is the size in bytes of the freed heap block?
`1) free(p1); p1 = NULL;` 32

→ Given the modified heap above, what is the size in bytes of the freed heap block?
`2) free(p4); p4 = NULL;` 24

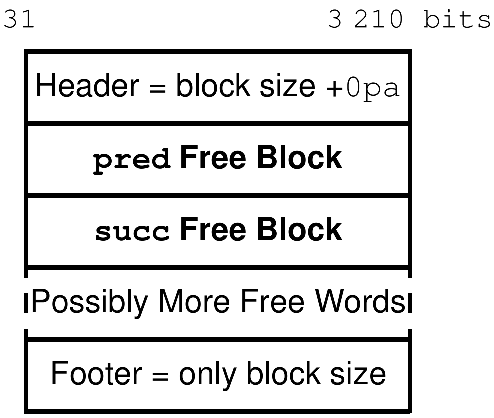
➢ Is coalescing done in a fixed number of steps (constant time)
or is it dependent on the number of heap blocks (linear time)?

Explicit Free List

- *An allocator using an explicit free list* only keeps a list of free blocks.

This explicit free list can be integrated into the heap by specifying a layout of free blocks only.

Heap Free Block Layout with Footer and Explicit Free List



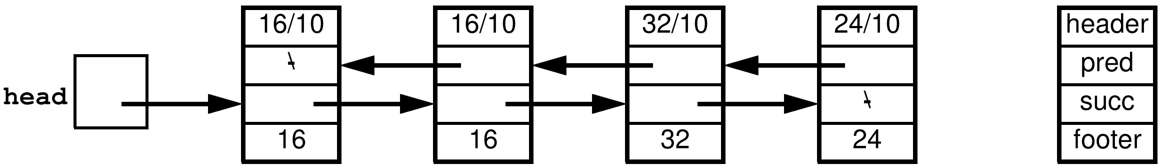
Free Block Links

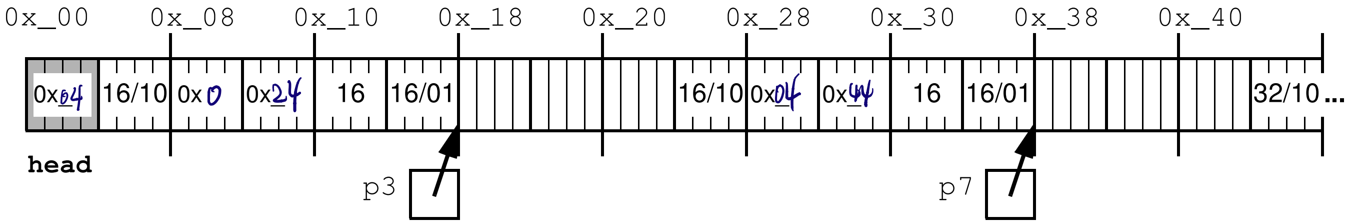
pred ADDR of previous free blocks.

succ ADDR of next free blocks.

→ Complete the addresses in the partially shown heap diagram below.

Free List as a Doubly-Linked List of Nodes **Node Layout**





→ Why is a footer still useful?

For fast coalescing with prev block.

→ Does the order of free blocks in the free list need to be the same order as they are found in the address space? No!

Copyright © 2016-2019 Jim Skrentny