Lecture 20:
Memory Allocation Implementation

CSE 29: Systems Programming and Software Tools

Aaron Schulman (Shalev)

UCSDCSE
Computer Science and Engineering

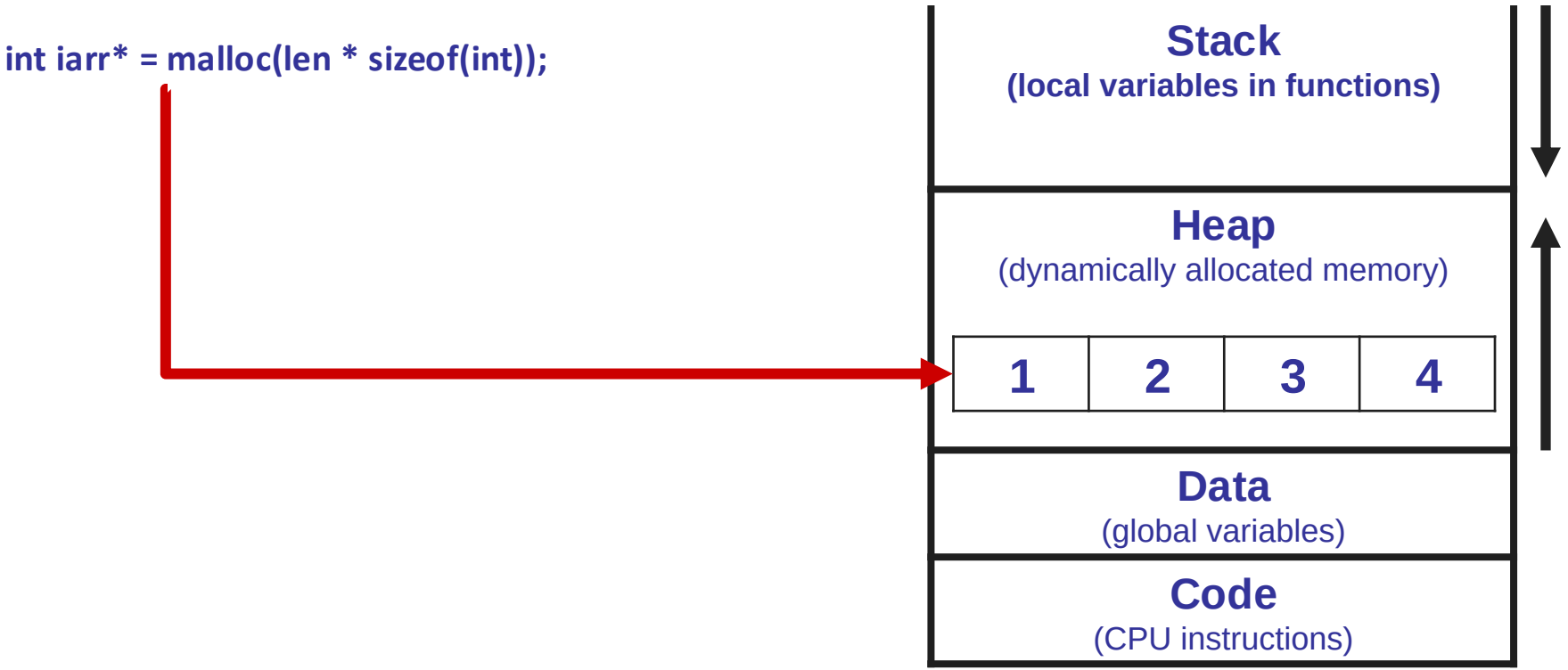
Announcements

- **Test 3** will *only* have coding; may be multi-part. No fill-in/conceptual
- **Test 2** grades soon (sorry!)
- **pa1-resubmit** is up, check Piazza post, make private Piazza post with questions
- **pa2-resubmit** is due “tonight” (Wednesday night, so tonight the day of lecture)

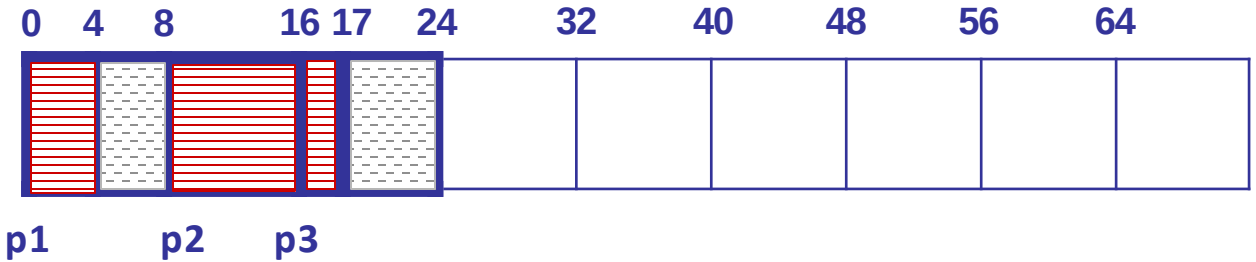
Logistics

- *Beginning a new phase of the class today:*
  - First half of the class was focused on **low-level programming in C**
  - Second half of the class is focused on **how a computer system works ("systems")**
    - How does memory allocation work?
    - How does information get stored on a disk?
    - Intro to OS: How multiple processes run at the same time

What does malloc() do?



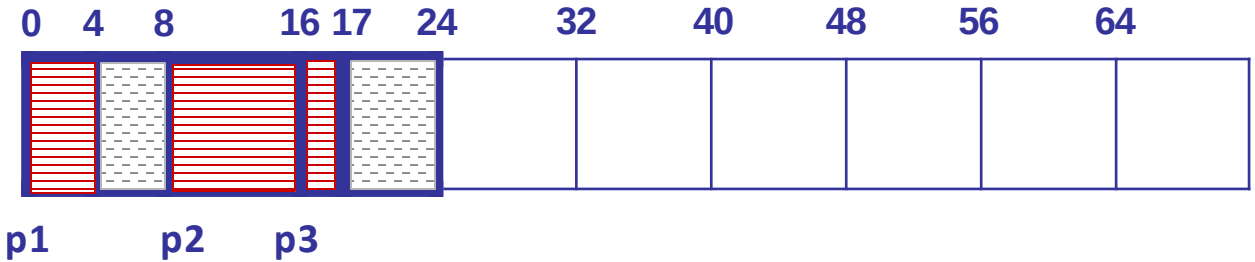
Malloc Implementation: Heap structure




```
int* p1 = malloc(4);
int* p2 = malloc(8);
int* p3 = malloc(1);
```

Alignment Requirement for 64-bit machine:
8 bytes (every heap allocation should be aligned to 8 bytes)

64-bit pointers are aligned at 8-byte boundaries

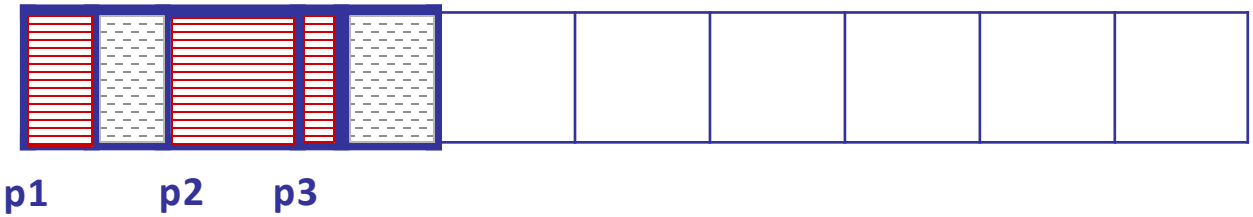



```
int* p1 = malloc(4); // 0x00
int* p2 = malloc(8); // 0x08
int* p3 = malloc(1); // 0x10
```

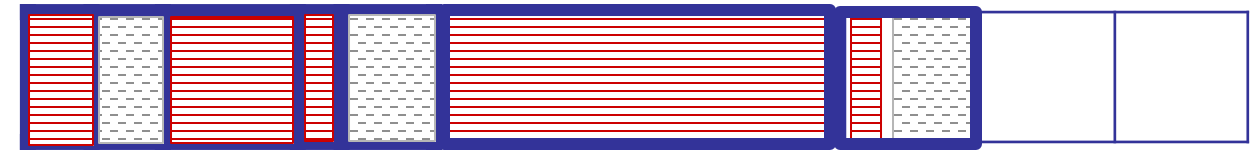
Malloc Implementation: Requirements

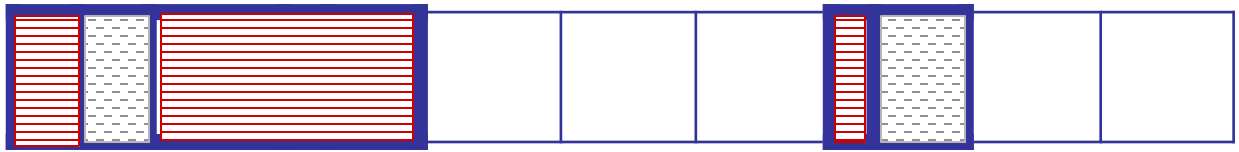
Requirements:

1. Only use the heap to manage the heap
2. Can not modify previously allocated blocks



When you free it leaves a hole in the heap





Free(big chunk)

How do we reuse these free spaces?

Goals of Malloc

- **Performance**
  - Speed of execution of malloc & free
    - » Think of this as allocation requests processed per second

- **Memory utilization efficiency**
  - Waste as little memory as possible when allocating new memory.
    - » Think of this as trying to avoid fragmentation of the heap where many memory regions between other regions are unusable because they are too small and scattered.

There is *no perfect tradeoff* between these two, just be good enough!

How does the Linux GNU Malloc work?

```
43    * Why use this malloc?
44
45      This is not the fastest, most space-conserving, most portable, or
46      most tunable malloc ever written. However it is among the fastest
47      while also being among the most space-conserving, portable and tunable
48      Consistent balance across these factors results in a good general-purp
49      allocator for malloc-intensive programs.
50
51      The main properties of the algorithms are:
52      * For large (>= 512 bytes) requests, it is a pure best-fit allocator,
53        with ties normally decided via FIFO (i.e. least recently used).
54      * For small (<= 64 bytes by default) requests, it is a caching
55        allocator, that maintains pools of quickly recycled chunks.
56      * In between, and for combinations of large and small requests, it doe
57        the best it can trying to meet both goals at once.
58      * For very large requests (>= 128KB by default), it relies on system
59        memory mapping facilities, if supported.
60
61      For a longer but slightly out of date high-level description, see
62         http://gee.cs.oswego.edu/dl/html/malloc.html
63
```