Announcements

- Midterm #2 is Today
- Program #4 on the web
- Reading
  - Notes (Today)
  - Notes (Tuesday)

Reminder of what we were doing a week ago

```
#include <stdio.h>
#include <unistd.h>

typedef struct _header {
    unsigned int size:29;
    unsigned int unusedBits:2;
    unsigned int allocated:1;
    struct _header *prev;
} header;

#define HEAP_INCR (1024 * 1024)
static header *heap;
```

Alternative: power of two sized space

- Maintain separate free lists of different sized objects
  - 8, 16, 32, 64, 128, 256, 512, 1024, 4096 bytes for example
- Never Split a block
- If a free list of a given size is empty
  - use sbrk to get more storage (some increment of size)
  - create a new free list of these items
- free
  - add back to free list for target size
  - if all items on a page (increment size) are free, release to common free pool

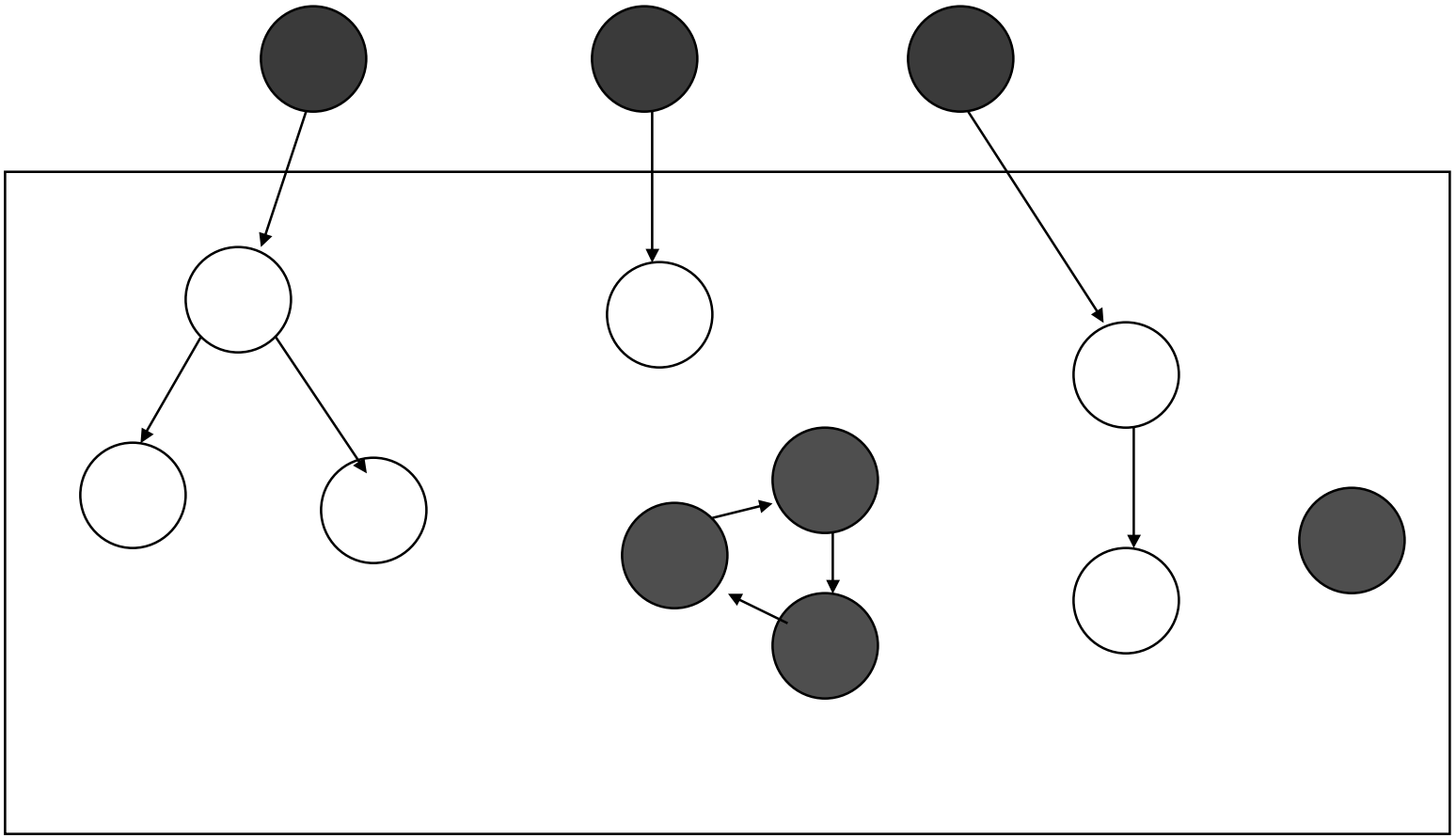
How to make heaps safer

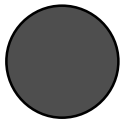
- aux tag storage
  - create additional (redundant) info elsewhere in heap
- preamble/trailer with known patterns
  - allocate 2 * Y bytes extra
  - fill out first and last Y bytes with a known pattern (non-zero)
  - return ptr Y bytes into region
  - on free (or when user requests)
    - verify y bytes at start and end haven't changed
- mcheck_pedantic(NULL);
  - from <mcheck.h>

Garbage Collection

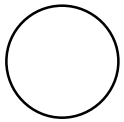
- What is Garbage?
  - memory the program can't get to anymore
  - Example:
    - ptr = malloc(40);
    - ptr = NULL;
- How to detect Garbage?
  - Need to know what is a pointer
    - know what everything is (Java)
    - guess anything that looks like a pointer is one (C)
      - any 32-bit quantity in globals, on stack, is assumed to be a pointer
      - anything in a memory region pointed to
  - Compute reachability graph
    - Follow all pointers

Reachability Graph

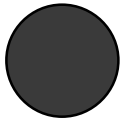




Not-reachable
(garbage)



Reachable



variables
(Inside the program)

Mark and Sweep Garbage Collection

- Mark Function
  - each allocated block has a "marked" field
  - isPtr returns header for ptr (if in heap) or null

```
void mark(ptr p) {
    ptr  b;
    b = isPtr(p);
    if (!b) return;
    if (b->marked) return;
    b->marked = 1;
    for (i=0; i < b->size; i++) {
        mark(b + i + 1);
    }
}
```

Mark & Sweep

- Sweep

```
void sweep(header *curr)  {
    while (curr->size) {
        if (curr->marked) {
                curr->marked = 0;
        } else if (curr->allocated) {
                free(curr+1);
        }
        curr = curr->next;
}
```

- Calling Mark and Sweep
  - Call when the allocate function is about to call sbrk

Garbage Collection

- Writing isPtr
  - need to be able to find the block header given a ptr into it
- Conservative Garbage Collector
  - something may look like a pointer, but not be one