CS61 Lecture 4

* pset clarification: malloc() calls m61_malloc()

  * add padding! after struct!

Alignment and Layout Rules

(1) first member law:

address of collection ≡ address of first member

(2) struct:

2nd, 3rd, subsequent members laid out in declaration order, no overlap, minimal padding, subject to alignment

(3) array rule:

elements laid out sequentially w/o gaps

(4) union rule:

```
union NAME {                NAME u;
    T1  x                   (uintptr_t) &u
    T2  y      stores           == (...) &u.x
    T3  z      one              == (...) &u.y
};                              == (...) &u.z
```

all member addr ≡ addr of union

* if you put x in the union, forgot you put a x and tried to read a T2 out, undefined behavior

(5) minimum rule

minimum size and alignment (no extra padding)

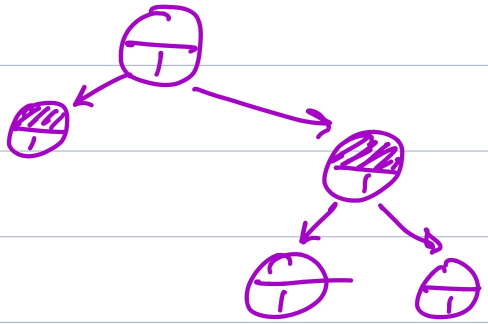
(6) malloc rule

every call to malloc that doesn't fail returns memory suitable for any alignment

allocate 1 byte → allocates 16 bytes

* tagged pointer representation

* red black trees



struct n {

T payload;

n* left_child;

n* right_child;

bool color;}

Pointer Arithmatic

$T\ a[N]$

$T*\ p = \&a[i]$

$T*\ q = \&a[i+1]$

$(q-p) \equiv 1$

$(p == q) \equiv 0$

$(p\ !\ q) \equiv 1$

$(p < q) \equiv 1$

arithmatic on pointers

arithmatic on addresses

$(\text{uintptr\_t})q - (\text{uintptr\_t})p \equiv \text{sizeof}(T)$

$\underset{\text{ptr}}{p} + \underset{\text{integer}}{x} \equiv \&p[x] \equiv \&a[i+x]$

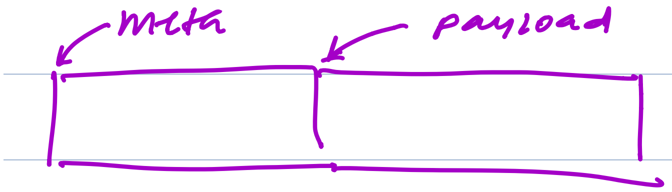
Rules for Forming Pointers

$T\ a[N]$

✓ for a pointer $\&a[i]$ iff $0 \le i \le N$

✓ to dereference a ptr $\&a[i]$ iff $0 \le i < N$

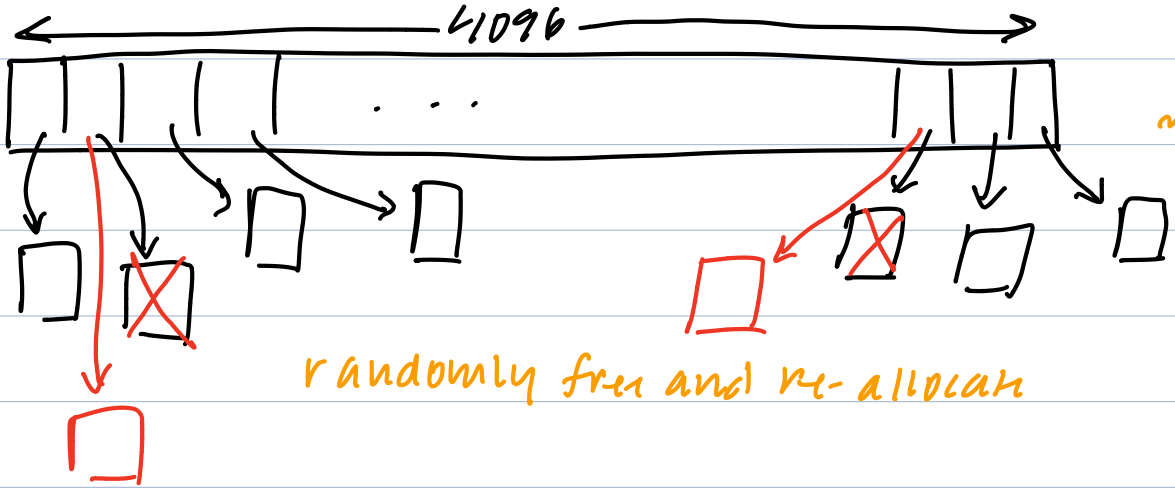
X form pointer that goes beyond the bounds of the array



```
(uintptr_t) x = (u... ) payload;
(uintptr_t) y = x - sizeof(M);
 M* m = (M*) y;
```

```
M* m = (M*) payload-1
```

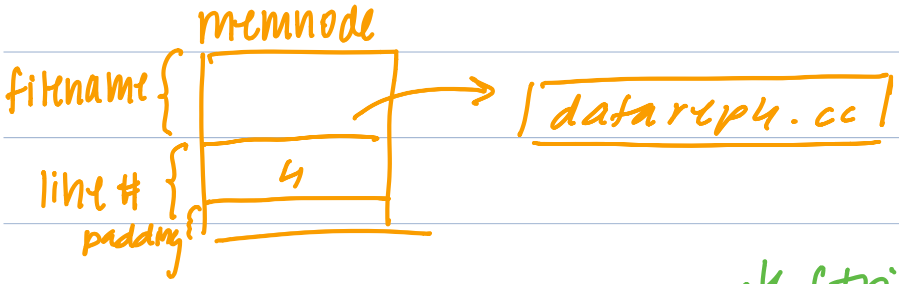
Memory Benchmark



~ 0.3 seconds for malloc

randomly free and re-allocate

* because vectors grow and shrink, they must be on the heap



* strings are const char*