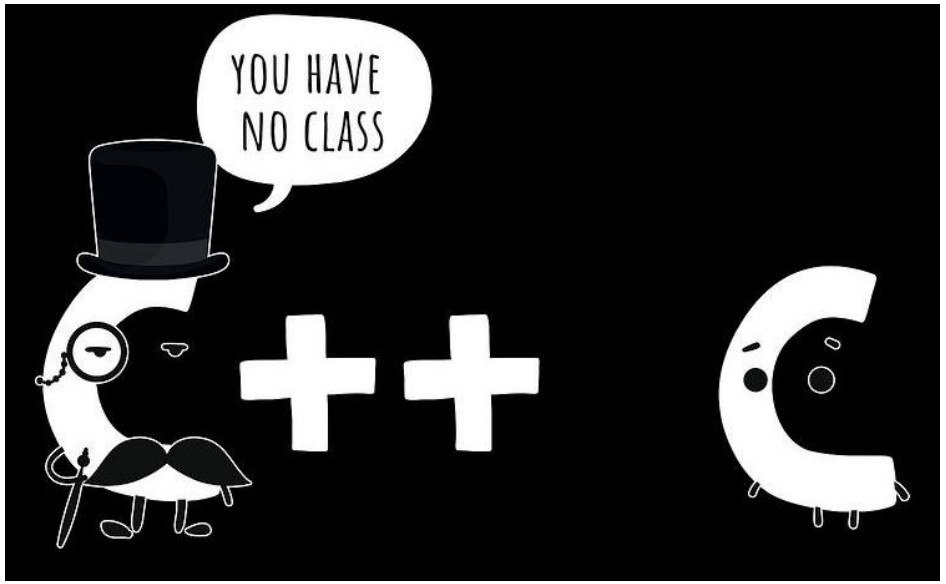
Structs & Alignment

CSE 351 Autumn 2023

Instructor:
Justin Hsia

Teaching Assistants:

Afifah Kashif	Malak Zaki
Bhavik Soni	Naama Amiel
Cassandra Lam	Nayha Auradkar
Connie Chen	Nikolas McNamee
David Dai	Pedro Amarante
Dawit Hailu	Renee Ruan
Ellis Haker	Simran Bagaria
Eyoel Gebre	Will Robertson
Joshua Tan	



https://pixels.com/featured/1-computer-programmer-funny-c-class-joke-noirty-designs.html

Relevant Course Information

- Lab 2 due tonight
- Lab 3 released next Monday (10/30)
  - A shorter lab, due Friday, 11/10
- hw13 due next Wednesday (11/1)

- **Take-home Midterm** (11/2 – 11/4)
  - Instructions will be posted on Ed Discussion
  - Gilligan's Island Rule: discuss high-level concepts and give hints, but not solving the problems together
  - We will be available on Ed Discussion (private posts only) and support hours to answer clarifying questions

Structs & Alignment

Lesson Summary (1/2)

- Structures
  - Allocate bytes for fields in order declared by programmer – can make choices to minimize memory allocations
  - Pad in middle to satisfy individual element alignment requirements (K)
  - Pad at end to satisfy overall struct alignment requirement (K_{max})

Lesson Summary (2/2)

- Terminology:
  - Structs: tags and fields, `.` and `->` operators
  - Typedef
  - Alignment, internal fragmentation, external fragmentation

- Learning Objectives:
  - Analyze the memory layout of a struct and minimize its impact on program memory usage.
  - Create, access, and modify array and struct elements in C.

- What lingering questions do you have from the lesson?

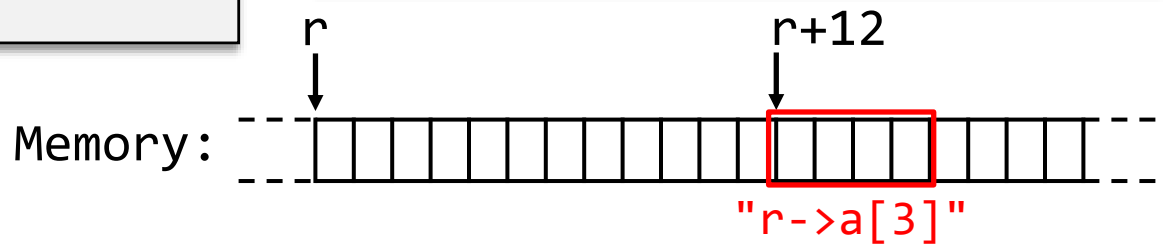
Structs & Alignment – Context

Struct Pointers

- Pointers store addresses, which all "look" the same
  - Lab 0 Example: struct instance `Scores` could be treated as array of `int`s of size 4 via pointer casting
  - A struct pointer doesn't *have* to point to a declared instance of that struct type
- Different struct fields may or may not be meaningful, depending on what the pointer points to
  - This will be important for Lab 5!

```
long get_a3(struct rec* r) {
  return r->a[3];
}
```

```
  movl  12(%rdi), %rax
  ret
```



Structs & Alignment – Practice

Group Work Time

- During this time, you are encouraged to work on the following:
  1) If desired, continue your discussion
  2) Work on the lesson problems (solutions at the end of class)
  3) Work on the homework problems

- Resources:
  - You can revisit the lesson material
  - Work together in groups and help each other out
  - Course staff will circle around to provide support

Practice Questions (1/2)

```
struct ll_node {
  long data;
  struct ll_node* next;
} n1, n2;
```

tag; fields; two instances; $K_{max}=8$

- How much space does (in bytes) does an instance of `struct ll_node` take? 16 B

- Which of the following statements are syntactically valid?
  - ✓ `n1.next = &n2;`
  - ✗ `n2->data = 351;`
  - ✓ `n1.next->data = 333;`
  - ✗ `(&n2)->next->next.data = 451;`

• for struct instances (inst), -> for struct pointers (ptr)

Practice Questions (2/2)

- Minimize the size of the struct by re-ordering the fields:

```
struct old {
  int i;
  short s[3];
  char *c;
  float f;
};
```

K: 4, 2, 8, 4 → $K_{max} = 8$

```
struct new {
  int    i;
  float  f;
  char * c;
  short  s[3];
};
```

could also switch these (internal vs. external frag)

  - What is the minimum size of `struct new`?

A. 22 bytes **B. 24 bytes**

C. 28 bytes D. 32 bytes