CMPSC 473
Operating Systems
Design & Construction

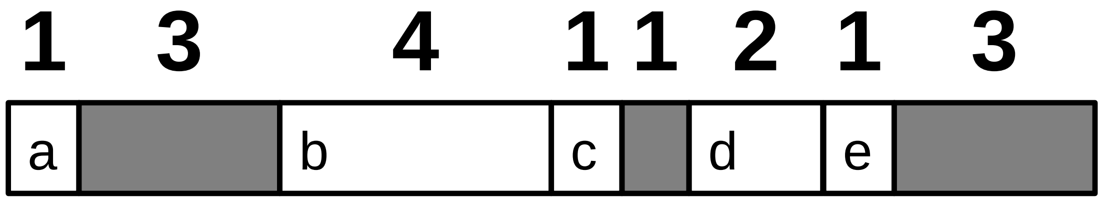
Instructor: Ruslan Nikolaev

The Pennsylvania State University

August 24, 2023 – Lecture 2

Structure alignment

```
struct {
  char a;
  int b;
  char c;
  short d;
  char e;
};
```



- Ordered according to declaration
- Alignment is determined by the ABI (Application Binary Interface) and CPU architectures: elements ***typically*** aligned to sizeof(element) or 8 bytes (64-bit CPUs), whatever is smaller

Structure alignment

- Each type has its own alignment requirement
- The compiler keeps track of the current alignment and inserts corresponding padding
- The very end also needs to be padded
  - e.g., consider an array of structure instances, each array entry must start at an appropriately aligned address

typedef

- Create your own type

Example:

```
typedef int ID;
typedef struct pt { int x; int y; } Point;
typedef int (*compare_fn)(int x, int y);
```

Benefits:

- Know what data you’re dealing with: ID vs int
- Easily change types: change ID int ▭ long
- Avoid typing struct: struct pt vs Point
- Avoid messy casts: (int (*)(int, int)) vs (compare_fn)

Header files

- Header files (.h) typically contain definitions (e.g., structs, function prototypes)
- Source files (.c) typically contain implementation

Example header file wrapper:

```
#ifndef FILE_NAME_H
#define FILE_NAME_H
Header content
#endif
```

- Header content only compiled once
- Prevents redefinitions from multiple includes

Defines & C preprocessor

`#define MY_CONSTANT 5`

- Replaces every MY_CONSTANT with 5

`#define MY_STRING hello`

- Replaces every MY_STRING with hello

`#if MY_CONDITION/#else/#endif`

- Tests MY_CONDITION and conditionally compiles code

`#ifdef DEF/#endif (#ifndef DEF/#endif)`

- If DEF is defined (not defined), conditionally compile code

Constants

- Bad style: magic numbers in code
- Better style: `#define CONSTANT 5`
- Even better: `static const int CONSTANT = 5;`
  - Includes type information

Const with pointers:

- Pointer to const data: `const int* p;`
- Const pointer to variable data: `int* const p;`
- Const pointer to const data: `const int* const p;`