

C Introduction

CS 2130: Computer Systems and Organization 1

Xinyao Yi Ph.D.
Assistant Professor

Announcements

- Homework 7 **due Monday at 11:59pm** on Gradescope
- Quiz 7 due Sunday
- Exam 2 next Friday

Pointers and Arrays

`*x` and `x[0]` are equivalent

- Pointer to single value and pointer to first value in array
- Treat array as pointer to the first value (lowest address)
- Indexing into array: `x[n]` and `*(x+n)`
 - If `x` is an `int *`, then `x+1` points to **next int** in memory
 - Adding 1 to pointer adds `sizeof( )` the type we're pointing to

Pointers

Consider the following code:

```
int x = 10;  
int *y = &x;  
int *z = y + 2;  
long w = ((long)z) - ((long)y);
```

Why is $w = 8$?

Other Types and Values

- Literal values - integer literals are implicitly cast
 - `unsigned long very_big = 9223372036854775808uL`
 - u for unsigned, L for long
- `enum` - named integer constants (in ascending order)
 - `enum { a, b, c, d=100, e };`
`int foo = e;`
- `void` - a byte with no meaning or "nothing"
 - Pointers: `void *p`
 - Return values: `void myfunction();`
- Casting - changing type, converting
 - Integer: zero- or sign-extend or truncate to space
 - Int to float: convert to nearby representable value
 - Float to int: truncate remainder (no rounding)

Structures

`struct` - Structures in C

- Act like Java classes, but no methods and all public fields
- Stores fields adjacently in memory (but may have padding)
- Compiler determines padding, use `sizeof()` to get size
- Name of the resulting type includes word `struct`

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

```
struct foo x;  
x.b = 123;  
x.c = 4;
```

Structure Literals

```
struct a {  
    int b;  
    double c;  
};
```

```
/* Both of the following initialize b to 0 and c to 1.0 */  
struct a x = { 0, 1.0 };  
struct a y = { .b = 0, .c = 1.0 };
```

typedef

typedef - give new names to any type!

- Fairly common to see several names for same data type to convey intent
- Ex: `unsigned long` may be `size_t` when used in sizes

- Examples:

```
typedef int Integer;
```

```
Integer x = 4;
```

```
typedef double ** dpp;
```

- Used with *anonymous structs*:

```
typedef struct { int x; double y; } foo;
```

```
foo z = { 42, 17.4 };
```

Struct Example

```
typedef struct {  
    long x;  
    long y;  
    long *array;  
    long length;  
} foo;
```

Struct Example

```
long sum2(foo *arg) {  
    long ans = arg->x;  
    for(long i = 0; i < arg->length; i += 1)  
        ans += arg->y * arg->array[i];  
    return ans;  
}  
  
sum2:  
    movq    (%rdi), %rax  
    movq    24(%rdi), %r8  
    testq   %r8, %r8  
    jle     .LBB1_3  
    movq    8(%rdi), %rdx  
    movq    16(%rdi), %rsi  
    xorl    %edi, %edi  
.LBB1_2:  
    movq    (%rsi,%rdi,8), %rcx  
    imulq   %rdx, %rcx  
    addq    %rcx, %rax  
    incq    %rdi  
    cmpq    %rdi, %r8  
    jne     .LBB1_2  
.LBB1_3:  
    retq
```

Struct Example

```
long sum1(foo arg) {  
    long ans = arg.x;  
    for(long i = 0; i < arg.length; i += 1)  
        ans += arg.y * arg.array[i];  
    return ans;  
}  
  
sum1:  
    movq    8(%rsp), %rax  
    movq    32(%rsp), %r8  
    testq   %r8, %r8  
    jle     LBB0_3  
    movq    16(%rsp), %rdx  
    movq    24(%rsp), %rsi  
    xorl    %edi, %edi  
.LBB0_2:  
    movq    (%rsi,%rdi,8), %rcx  
    imulq   %rdx, %rcx  
    addq    %rcx, %rax  
    incq    %rdi  
    cmpq    %rdi, %r8  
    jne     .LBB0_2  
.LBB0_3:  
    retq
```