

Structs

Ray Seyfarth

August 5, 2011

Structs in C

- A struct is a compound object

```
struct Customer {  
    int  id;  
    char name[64];  
    char address[64];  
    int  balance;  
};
```

- We can allocate a `Customer` if we know the size

```
mov     rdi, 136           ; size of a Customer  
call   malloc  
mov     [c], rax          ; save the address
```

Filling in a C struct

- We can fill in the parts given their offsets

```
mov     [rax], dword 7    ; set the id
lea     rdi, [rax+4]      ; name field
lea     rsi, [name]       ; name to copy to struct
call    strcpy
mov     rax, [c]
lea     rdi, [rax+68]     ; address field
lea     rsi, [address]   ; address to copy
call    strcpy
mov     rax, [c]
mov     edx, [balance]
mov     [rax+132], edx
```

Assembly struct

- Using the `yasm` `struc` pseudo-op we can define a `Customer`

```
        struc    Customer
id      resd    1
name    resb    64
address resb    64
balance resd    1
        endstruc
```

- `id`, `name`, `address` and `balance` are globals
- You could not have `id` in 2 structs
- It's almost the same as doing 4 equates
- The size is `Customer_size`

Assembly struct

- One alternative is to prefix field names with dots

```
        struc    Customer
.id     resd    1
.name   resb    64
.address resb    64
.balance resd    1
        endstruc
```

- Then you would have to use `Customer.id`
- Another alternative is to use an abbreviated prefix

```
        struc    Customer
c_id     resd    1
c_name   resb    64
c_address resb    64
c_balance resd    1
        endstruc
```

Program to allocate and fill a struct - data segment

```
        segment .data
name    db      "Calvin", 0
address db      "12 Mockingbird Lane",0
balance dd     12500
        struc   Customer
c_id    resd    1
c_name  resb    64
c_address resb  64
c_balance resd  1
        endstruc
c       dq      0           ; to hold a Customer pointer
```

Program to allocate and fill a struct - part of text segment

```
mov     rdi, Customer_size
call   malloc
mov     [c], rax      ; save the pointer
mov     [rax+c_id], dword 7
lea     rdi, [rax+c_name]
lea     rsi, [name]
call   strcpy
mov     rax, [c]      ; restore the pointer
lea     rdi, [rax+c_address]
lea     rsi, [address]
call   strcpy
mov     rax, [c]      ; restore the pointer
mov     edx, [balance]
mov     [rax+c_balance], edx
xor     eax, eax
```

Alignment problems

- Suppose you increase the size of the `c_address` array to 65
- C would make the offset of `balance` be 136
- `yasm` would define the offset as 133
- The goal is to be C compatible
- Also C would have `sizeof(Customer)` as 140
- `Customer_size` would be 137
- C aligns each field and makes the size of a struct appropriate for aligning each data item properly if we allocate an array of structs
- We need to use `align` in the struct

```
        struc    Customer
c_id    resd     1
c_name  resb    64
c_address resb   65
        align    4
c_balance resd   1
        endstruc
```

Allocating a slightly more complex array of customers

```
        segment .data
        struc   Customer
c_id    resd    1      ; 4 bytes
c_name  resb   65     ; 69 bytes
c_address resb  65     ; 134 bytes
        align  4      ; aligns to 136
c_balance resd  1     ; 140 bytes
c_rank  resb   1     ; 141 bytes
        align  4      ; aligns to 144
        endstruc
customers dq    0
        segment .text
mov     edi, 100 ; for 100 structs
mul     edi, Customer_size
call    malloc
mov     [customers], rax
```

Printing an array of customers

```
format      segment .data
            db      "%s %s %d",0x0a,0
            segment .text
            push   r15
            push   r14
            mov    r15, 100          ; counter saved through calls
            mov    r14, [customers]; pointer saved through calls
more        lea    edi, [format]
            lea    esi, [r14+c_name]
            lea    rdx, [r14+c_address]
            mov    rcx, [r14+c_balance]
            call   printf
            add    r14, Customer_size
            sub    r15, 1
            jnz   more
            pop    r14
            pop    r15
            ret
```