# Hack postgres Source Code: Vol I

## Chapter 1 : C & Rust

### 1.35 Structure Padding & Packing - C

C invokes padding implicitly while allocating memory for structure variables.

For example -

```
#include<stdio.h>

// struct-line-1
struct emp {

    int id;
    char is_terminated;
    char is_active;

};

int main() {

    // main-line-1
    struct emp john =  { .id = 1, .is_active = 'Y', .is_terminated = 'N'};

    // main-line-2
    printf("%lu", sizeof(john));

}
```

`struct-line-1` : Creating type `struct emp` with some primitives.

`main-line-1` : In C, structure data is stored in contiguous memory locations. Pause a momemnt... Umm.. this is the time that we need to know some little bit of info about processor in our PCs. You know, processor is the that handles execution of machine code generated by C compiler from our C Code. Processor in our PC doesn't read one `byte` at time from memory, it reads one `word` at a time. However, definition of `word` changes based on whether the PC's architecture is 32 bit and 64 bit. If it is 32 bit, one `word` equals 4 bytes and 8 bytes for 64 bit. Time taken to read each `word` is called as one `cpu cycle` . Even though it feels out of the way at this point, let me tell you one more thing, we initialized structure variable(in other words structure object `john` ) with `designated initialization` style. In this way, we can allocate values using `.(variable_name)` in curly braces. We don't have to follow order of variable in sturct type. If we don't use `.(variable_name)` while allocating values, we should allocate according to the order in our structure type. Hmm.. back to

our processor.. when C come across `main-line-1` , it allocates `1 byte` for the `char` variable `is_terminated` , `1 byte` for another `char` variable `is_active` and `4 bytes` for `int` variable `id` . Totally, it fills 6 bytes in contiguous memory locations. You might wonder which memory area's memory locations? - stack or heap. As `john` is automatic variable/local variable and automatic variables are stored in stack, this memory allocation happens in stack only. We will see about heap in coming days. When processor(consider 32 bit) tries to read, it has to use 2 CPU cycles. Because, processor can only read 4 bytes and we have 6 bytes. It can't just split `int` variable bytes and takes them in first cycle. Instead, it adds two empty bytes next to second `char` variable and pushes `int` variable 2 bytes away so that it can read that `int` variable in next CPU cycle. Of course, this is all happens implicitly and people call it this process as `Structure Padding` . Because of structure padding, there is wastage of memory.

`main-line-2` : In C, there is `sizeof` operator which emits size of any type. That emitting value is of type `unsigned int` so we used `lu` as format specifier. We get 8 bytes as size due to structure padding.

By using `#pragma pack(1)` preprocessor we can overcome this wastage irrespective of order of types in struct types. This is called `Structure Packing` . After using this one, one can witness 6 bytes as size.