Hack `postgres` Source Code: Vol I

Chapter 1 : C & Rust

1.36 Unions - C

Unions are same as structures in notation except `union` keyword is used instead of `struct` keyword.

But, unlike structure, `union` members share same memory location.

Size of memory for `union` is decided by the members in union. Member having more size is used to decide size of memory.

For example,

```c
#include<stdio.h>

/*
* `sample_union` union type is created
* with int type and char type members.
* This union gets 4 bytes of memory.
* Size defined by member types in union.
* As int has more size than char,
* int's size is used to calculate union memory size.
*/
union sample_union {
    int emp_id;
    char is_active;
};

int main() {

    // Initialization of union
    // We can only initialize first member of union
    // as per C-89 style using curly braces.
    union sample_union emp = {1};
    emp.is_active = 'A';

    printf("%d", emp.emp_id);
    printf("%c", emp.is_active);
}
```

Above code results in `65 and A` as output.Its because... C stores `1` in 4 bytes of memory (32 bits) as `00000000 00000000 00000000 00000001` . As 1 is small number, first byte is enough to represent 1.

When C wants to store 'A' knowing it is from union, it starts to store from the first byte again. ASCII equivalent of A is 65 and 65 is stored into avaialable memory. So, we end up having following information in our 4 bytes of memory.

```
00000000 00000000 00000000 01000001
```