Hack `postgres` Source Code: Vol I

Chapter 1 : C & Rust

1.1 Enumeration Types in C & Rust.

`postgres` 's `initdb` binary code has enums (enumeration types). So, we need to know them.

`enums` are custom types like `structs` in C and Rust.

`enums` are used to improve readability of code.

C - Enums

- Enum values in C are associated with integer values only. If we don't pass any integer value, compiler assigns value.
- Enums can be created within or out of main function.

```c
#include<stdio.h>

/*
* Enum type creation for two weeks - Sunday and Monday.
* We mapped `0` to the enum value `Sunday`. Here, enum value `Sunday`
* is not string. It is just label/value used in enum type.
* As Monday is not assigned with any integer value, Compiler checks
* previous enum value's integer and map to `Monday' by incrementing
* it to 1. So, Monday value will be 1.
*/
enum Days { Sunday = 0, Monday };

int
main() {

    /*
    * Declaring a variable with enum type `enum Days` with the
    * identifier `day'.
    */
    enum Days day;

    /*
    * Adding a value to enum variable `day`
    * `Sunday` is not string but a allowed enum value to be assigned
    *  to `day` enum variable.
    */
    day = Sunday;

    /*
    * Compiler takes value in `day` variable i.e., Sunday.
    * Sunday corresponds to `0` in defined enum type. So, compiler
    * converts enum value to integer values and compares like `0 == 0`
    */
    if (day == Sunday) {
        printf("It's Sunday!");
      }

}
```

- If we want to implement above program in arrays/structs, it increases usage of indexes(in case of arrays) and variables(in case of structs.. to maintain enum value and associated integer value).

Rust - Enums

- Unlike C, we can map any type to enum value in rust.

```
// Creating enum type
enum Weeks {
     Sunday,
     Monday,
    }; // No need of semicolon

fn main() {

    let first_week: Weeks = Weeks::Sunday;
    let second_week: Weeks = Weeks::Monday;
    /*
    * We use match statement in rust
    * If passed expression `first_week` is matched with any patterns
    * like `Weeks::Sunday` or `Weeks::Sunday`, it executes code after
    * `=>`
    */
    match first_week {
        Weeks::Sunday => {
        println!("Its Sunday"); }
        Weeks::Monday => {
        println!("Its MondaY"); }
   }

   match second_week {
        Weeks::Sunday => {
        println!("Its Sunday"); }
        Weeks::Monday => {
        println!("Its MondaY"); }
   }

}
```