Hack postgres Source Code: Vol I

Chapter 1 : C & Rust

1.1 Ownership System - Slice Type in Rust. Making C & postgres Sleep for Sometime.

Rust allows to have sliced references.

We can actually create a reference to slice of string in rust and passes it to functions/variables.

```
fn main() {

    //main-line-1
    let _str: String = String::from("John Doe");

    //main-line-2
    let _str_slice: &str = send_first_letter_first_name(&_str[0..4]);

    //main-line-3
    println!("{}", _str_slice);

}

fn send_first_letter_first_name(_s: &str) -> &str {

    &_s[0..1]

}
```

`main-line-1` : rust organizes the string object "Jon Doe" in stack and heap memory in RAM.

`main-line-2` : rust uses `[start_index..end_index]` notation to slice the string object data by index. String objects in rust follow `0` index convention in which first element in string object start at index - `0` . `start_index` in slice notation is inclusive and `end_index` is exclusive. when rust comes across `&_str[0..4]` , it goes to heap memory location of string object. String object is composed by characters and they are scattered across several memory locations in heap (rust uses 4 bytes to store `char` data type. We can imagine 1 byte as 1 memory location). Later, rust finds the memory location of `start_index` element and moves continuosly through remaining memory locations until the `end_index-1` element. Rust stores string object's individual characters contiguosly in heap memory. Once, it gets the slice of memory location, it passes that sliced memory location address/ reference to called function `send_first_letter_first_name` . After that,

called function `send_first_letter_first_name` slices that reference again using `start_index` and `end_index`. We return that sliced memory location address/reference to calling function.

`main-line-3: println!` macro prints the data in memory location held by `_str_slice` slice type/sliced reference.

String literals in rust are slice types.