Hack `postgres` Source Code: Vol I

Chapter 1 : C & Rust

1.32 String Literals & Strings - C

String literals are contstants/immuatble in C. We can modify its individual characters in it. It is always enclosed in double quotes.

String literals are referred by a pointer of char type.

```
char *string_ptr = "Hello";
```

When C sees that above statement, it makes the pointer of type `string_ptr` to hold the address of the first character of string literal. In this case, the pointer holds the address of `H`. And, also the string is stored in memory having additional character at the end, called as null character `\0`.

```
#include <stdion.h>

int main() {

    char *s = "Hello";

    printf("%s", s);

}
```

Note: If we use `*s` in `printf`, its like dereferencing what is there in pointer `s` which is nothing but address of first character and reslut would be just `H`. By passing just `s` without `*`, C making `printf` to traverse until it sees null character - `\0` that results in displaying entire string on console when we run the executable.

On the other hand, strings are mutable in C. These are kind of one dimensional character arrays. So, we can apply all types of array operations on them like modifying individual characters.

Strings can be initialized in different ways and C handles those different ways in its own style.

`char[6] char_str = "Hello";` -> C stores five characters in memory by attaching null character at the end. We need to make sure to have the size to accomodate that extra null character while initializing strings.

`char[6] char_str = {'H','e', 'l', 'l', 'o', '\0'};` -> C stores as is.

`char[] char_str = "Hello";` -> C finds size `6` automatically in this case and stores the string in memory accordingly.

```
#include<stdio.h>

int main() {

    char s[] = "Hello";

    printf("%s", s);

}
```