# Hack `postgres` Source Code: Vol I

## Chapter 1 : C & Rust

### 1.7 Function `get_progname` in `postgres`'s `main.c`

Pi was getting flashes of white board full of conditionals such as if-else if-else and loops in Go's notebook out of nowhere same as Go at his car in restaurant's underground parking lot.

He tangled his fingers in her hair, massaging the base of her skull and keeping her from moving away while kissing at the car door.

She hadn't been feeling this way for weeks. Breathless, body burning for release, unable to focus on anything but the pleasure building between her legs. Her nipples tightened, and his gaze fell to her breasts.

Nobody was watching them. No body saw them in the dark parking area.

Suddently, Go's phone rang continuosly. She looked at it and found that it was her friend Rachel. She picked it up and moved herself away from Pi's kiss.

"Where are you?, Go" asked Rachel worriedly.

It's the first time that Go was late since they moved into a new room. So, Rachel was getting worried about her friend.

"At the dinner and on the way back to home" replied Go casually.

After knowing Go was fine, Rachel hung up.

Pi and Go got into the car and he dropped her at the room which was near by college.

(Next Day)

Pi was coming into the class room gazing at back of Go's pink tshirt. She was scrolling through the notes that were taken in previous classes sitting on wooden chair in the middle of third row on the left of classroom. She seemed no different as if nothing was happened in last night.

Pi fired up his laptop which was placed on a wooden booth placed closest to white board and right of the classroom and its projector. Then, he wrote "Functions" in the notepad in his laptop which got projected on white wall to make it visible for Go.

"Today, we are gonna discuss about functions" said Pi in a voice as if he taught that million times.

"Ok" said Go with a grin on her face.

They were reminiscing their last night encounter silently.

Pi started to write about functions in his notepad.

Pi's notepad -

### C:

```c
#include<stdio.h>

int sum(int a, int b); // function prototype

int
main() {

  int a = 10, b = 20, s;

  s  = sum(a,b);

  printf("%d", s);

  return 0;
}

int sum(int a, int b) {

    return a+b; // returning a value from function

}
```

### Rust:

```rust
fn main() {

    let a: i32 = 10;
    let b: i32 = 20;
    let c: i32;

    c = sum(a, b);

    println!("{}", c);

}

fn sum(a: i32, b: i32) -> i32 {

    a+b // returning value from function

}
```

"C and Rust have different syntax for functions. C needs function prototype which is also called as function declaration that can be identified with semicolon at the end of it. This is used to inform compiler about the existence of that function. On the other hand function definition contains function's implementation that you can see at the bottom of the main." uttered Pi in his determined voice.

"what are those bunch of types around function prototype and definition" asked Go.

"In C, a function is something that is used to do some task and can be invoked from some other function. In our case that is main function. main function is special function in C which is not invoked from any other function. A function which calls another function is called as calling function and the function which is being called is labelled as called function. In our example, the calling function is main and the called function is sum. Generally, calling function passes values to called function and called functions return values as a response. But, it is not necessary to pass values from calling function and get values from called functions. In our example, calling function is sending two integers to called function, so it shold be designed to receive only integeres. That is why, we had int a and int b in function sum parenthesis. As sum function is returning sum of a and b which is also an integer, we place int before sum to indicate that the sum function is desinged to return integer. On the whole, we are using those bunch of ints in our example to show the working of our function in high level."

Go looked up after writing some key points and nodded in a way that she got answer for her question.

"Rust also follows the same principles as in C mostly. But, Rust doesn't require any prototypes and compiler automatically finds whereabouts of the functions used in calling functions. Placement of types around function also different from C as shown in our example. But, rust requires types around function. Rust doesn't use return keyword and uses a code line that doesn't need semicolon. In programming languages, expressions are the ones which return values and statements are the ones which doesn't return values. Rust uses this concept and doesn't use the return keyword even though it is permissible to use return key word. To make the understanding clear, we can say that the value is returned from rust function if we don't use any semicolon after it and return keyword before it."

"Any doubts?" asked Pi.

"No" said Go.