Write a function that returns the length of a string.

Prototype: int \_strlen(char \*s);

ChatGPT

Of course! Here's a simple function that returns the length of a string:

c

Copy code

int \_strlen(char \*s) {

int length = 0;

while (\*s) { // While the character pointed to by 's' is not the null terminator

length++;

s++; // Move to the next character

}

return length;

}

You can use the function \_strlen just like the standard library's strlen function:

c

Copy code

#include <stdio.h>

int \_strlen(char \*s);

int main() {

char str[] = "Hello, World!";

printf("Length of string: %d\n", \_strlen(str));

return 0;

}

This will output:

c

Copy code

Length of string: 13

User

int \_strlen(char \*s) {

int length = 0;

while (\*s) { // While the character pointed to by 's' is not the null terminator

length++;

s++; // Move to the next character

}

return length;

}

I need to know more about while (\*s)

ChatGPT

Certainly! The line while (\*s) is a concise way of writing a loop that continues as long as the character pointed to by the pointer s is not the null terminator ('\0'), which marks the end of a C-style string.

In C, every string is terminated by a null character ('\0'). The value of '\0' is 0. The condition while (\*s) will be true (non-zero) for any character other than the null terminator.

Let's break down the loop:

Initially, s points to the first character of the string.

\*s dereferences the pointer, which means it gives us the actual character s is pointing to.

The while loop checks if this character is not the null terminator. If it's not, it enters the loop body.

Inside the loop, length is incremented because we've encountered a character that's part of the string.

The pointer s is then incremented to point to the next character of the string.

The loop then checks the next character in the same way.

This continues until s points to the null terminator, at which point \*s evaluates to 0, the loop condition becomes false, and the loop exits.

To give a clearer understanding, consider the string "abc". Here's a step-by-step representation of how the loop progresses:

rust

Copy code

s -> "abc\0"

\* ^

\* This is the first character. It's 'a', which is not '\0', so the loop continues.

s -> "abc\0"

\* ^

\* Now, 's' points to 'b'. Again, it's not '\0', so the loop continues.

s -> "abc\0"

\* ^

\* Now, 's' points to 'c'. Still not '\0', so the loop continues.

s -> "abc\0"

\* ^

\* Finally, 's' points to '\0'. The loop exits because \*s is now 0 (false in boolean context).

In summary, while (\*s) is a common C idiom for iterating over each character in a string until reaching the null terminator.