

Certainly! Let's walk through an example using the `_strstr` function with the given strings:

```
char the_big_str[] = "Hello, World!";  
char the_little_str[] = "World";
```

Now, let's call the `_strstr` function:

```
char* result = _strstr(the_big_str, the_little_str);
```

1 Initial state:

2 `x = 0, y = 0, z = 0`

3 First iteration of the outer loop (while (the_big_str[x] != '\0')):

4 `x = 0`

5 Inner loop (while (the_little_str[y] == the_big_str[z] && the_little_str[y])):

6 `y = 0, z = 0`

7 Comparison: 'W' == 'H' (no match), so the inner loop breaks.

8 Second iteration of the outer loop:

9 `x = 1`

10 Inner loop:

11 `y = 0, z = 1`

12 Comparison: 'W' == 'e' (no match), so the inner loop breaks.

13 Third iteration of the outer loop:

14 `x = 2`

15 Inner loop:

16 `y = 0, z = 2`

17 Comparison: 'W' == 'l' (no match), so the inner loop breaks.

...and so on.

1 Seventh iteration of the outer loop:

- 2 `x = 6`
- 3 Inner loop:
 - 4 `y = 0, z = 6`
 - 5 Comparison: `'W' == 'W'` (match)
 - 6 Inner loop continues:
 - 7 `y = 1, z = 7`
 - 8 Comparison: `'o' == 'o'` (match)
 - 9 Inner loop continues:
 - 10 `y = 2, z = 8`
 - 11 Comparison: `'r' == 'r'` (match)
 - 12 Inner loop continues:
 - 13 `y = 3, z = 9`
 - 14 Comparison: `'l' == 'l'` (match)
 - 15 Inner loop continues:
 - 16 `y = 4, z = 10`
 - 17 Comparison: `'d' == 'd'` (match)
 - 18 Inner loop continues:
 - 19 `y = 5, z = 11`
 - 20 Comparison: `'\0' == '\0'` (match)
 - 21 Substring found:
 - 22 The inner loop completes, and the entire `the_little_str` is found starting from index 6 in `the_big_str`.
 - 23 The function returns a pointer to the beginning of the substring within the original string (`return (the_big_str + x)`).
 - 24 Result:
 - 25 `result` now points to the position of the substring in `the_big_str`.
 - 26 Printing the result: `"Substring found at position: 6"`

So, in this example, the `_strstr` function correctly finds the substring "World" in the string "Hello, World!" starting at position 6.

[explain the code](#)

Certainly! Let's break down the `_strstr` function:

```
char *_strstr(char *the_big_str, char *the_little_str)
{
    int x = 0, y = 0, z = 0;

    while (the_big_str[x] != '\0')
    {
        y = 0;
        z = x;

        // Inner loop compares characters in the_little_str and the_big_str starting from the current position
        while (the_little_str[y] == the_big_str[z] && the_little_str[y])
        {
            y++;
            z++;
        }

        // Check if the entire substring was found
        if (the_little_str[y] == '\0')
        {
            return (the_big_str + x);
        }

        x++;
    }

    return (NULL);
}
```

Here's a step-by-step explanation:

- 1 Initialization:
- 2 `x` is the index for iterating through `the_big_str`.
- 3 `y` is the index for iterating through `the_little_str`.
- 4 `z` is used to track the position in `the_big_str`.

5 **Outer Loop** (`while (the_big_str[x] != '\0')`):

6 The outer loop iterates through each character in `the_big_str` until a null terminator is encountered.

7 **Inner Loop** (`while (the_little_str[y] == the_big_str[z] && the_little_str[y])`):

8 The inner loop compares characters in `the_little_str` and `the_big_str` starting from the current position (`z`).

9 It continues as long as characters match and the end of `the_little_str` has not been reached.


10 **Substring Match:**

11 If the inner loop completes (i.e., the entire `the_little_str` is found), the function returns a pointer to the beginning of the substring within the original string (`return (the_big_str + x)`).

12 **Result:**

13 If the substring is not found at the current position, the outer loop continues to the next character in `the_big_str`.

14

From ChatGPT, designed with  FancyGPT