What's difference between char s[] and char *s in C?

Consider below two statements in C. What is difference between two?

```
char s[] = "geeksquiz";
char *s  = "geeksquiz";
```

Below are the key differences:

Char a[10] = "geek";	Char *p = "geek";
1) a is an array	1) p is a pointer variable
2) sizeof(a) = 10 bytes	2) sizeof(p) = 4 bytes
3) a and &a are same	3) p and &p aren't same
4) geek is stored in stack section of memory	4) p is stored at stack but geek is stored at code section of memory
5) char a[10] = "geek"; a = "hello"; //invalid > a, itself being an address and string constant is also an address, so not possible.	5) char *p = "geek"; p = "india"; //valid
6) a++ is invalid	6) p++ is valid
7) char a[10] = "geek"; a[0] = 'b'; //valid	7) char *p = "geek"; p[0] = 'k'; //invalid > Code section is r- only.

The statements '**char s[] = "geeksquiz"**' creates a character array which is like any other array and we can do all array operations. The only special thing about this array is, although we have initialized it with 9 elements, its size is 10 (Compiler automatically adds '\0')

```
#include <stdio.h>
int main()
{
    char s[] = "geeksquiz";
    printf("%lu", sizeof(s));
    s[0] = 'j';
    printf("\n%s", s);
    return 0;
}
```

Output:

```
10
jeeksquiz
```

The statement '**char *s = "geeksquiz"**' creates a string literal. The string literal is stored in read only part of memory by most of the compilers. The C and C++ standards say that string literals have static storage duration, any attempt at modifying them gives undefined behavior.
s is just a pointer and like any other pointer stores address of string literal.

```
#include <stdio.h>
int main()
{
    char *s = "geeksquiz";
    printf("%lu", sizeof(s));

    // Uncommenting below line would cause undefined behaviour
    // (Caused segmentation fault on gcc)
    //  s[0] = 'j';
    return 0;
}
```

Output:

```
8
```

Running above program may generates a warning also "warning: deprecated conversion from string constant to 'char*'". This warning occurs because s is not a const pointer, but stores address of read only location. The warning can be avoided by pointer to const.

```
#include <stdio.h>
int main()
{
    const char *s = "geeksquiz";
    printf("%lu", sizeof(s));
    return 0;
}
```

This article is contributed by Abhay Rathi. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

Recommended Posts:

What is the difference between "char a" and "char a[1]"?
Difference between const char *p, char * const p and const char * const p
char* vs std:string vs char[] in C++
What is the difference between single quoted and double quoted declaration of char array?
Wide char and library functions in C++
Convert string to char array in C++
C program to find the size of int, float, double and char
Web 1.0, Web 2.0 and Web 3.0 with their difference
What is the difference between GUI and CUI?
Difference between C and C++
Difference between ++*p, *p++ and *++p
What's difference between MMU and MPU?
Difference between scanf() and gets() in C
Difference between C and Python

Difference Between HTML and ASP

Article Tags : C Difference Between C Array and String cpp-string

Practice Tags : C

6

To-do Done

2.9

Based on **24** vote(s)

Feedback/ Suggest Improvement Add Notes Improve Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Previous

What's difference between Microcontroller (µC) and Microprocessor (µP)?

Next

How to check whether a number is in the range[low, high] using one comparison ?

Most popular in C

- Dividing a Large file into Separate Modules in C/C++, Java and Python
- Difference between scanf() and gets() in C
- C program to store Student records as Structures and Sort them by Name
- Program to Reverse a String using Pointers
- Inline function in C

Most visited in Difference Between

- Difference between SQL and NoSQL
- Differences between IPv4 and IPv6
- Logical vs Physical Address in Operating System

Stack vs Heap Memory Allocation

Difference between HashMap and HashSetv