Using fgets() with char* type

Asked 9 years, 4 months ago Modified 9 years, 4 months ago Viewed 20k times

7

I have a simple question about using fgets() with char* string.

```
....
char *temp;
FILE fp=fopen("test.txt", "r");

fgets(temp, 500, fp);
printf("%s", temp);
....
```

This code didn't work well.

But after I modified `char *temp` to `char temp[100];` , the code worked well as I intended.

What is the difference between those two?

When I googled it, some said that memory must be allocated to `char *` using malloc()...

But I couldn't understand it.

c string char fgets

Share Improve this question Follow

asked Nov 25, 2013 at 16:33

user3033077
69 1 1 4

2 If you don't allocate memory what will the pointer point to ?To remember this immediately assign NULL to pointer after declaration.Same holds when you free a pointer's memory.Assign NULL there also after deletion. No memory allocated pointer is a very common mistake and gives you the dreaded segmentation fault. – Rafed Nole Nov 25, 2013 at 18:44

3 Answers

Sorted by: Highest score (default)

6

char *temp is only a pointer. At begin it doesn't points to anything, possibly it has a random value.

fgets() reads 500 bytes from fp to the memory addresse, where this temp pointer points! So, it can overwrite things, it can make segmentation faults, and only with a very low chance will be

work relativale normally.

But char temp[500] is a 500 bytes long array. That means, that the compiler does the allocation on the beginning of your process (or at the calling of your function). Thus this 500 bytes will be a useable 500 bytes, but it has a price: you can't reallocate, resize, free, etc. this.

What the google wants from you, is this:

```
char *temp = (char*)malloc(500);
```

And a

```
free(temp);
```

after you don't need this any more.

Share Improve this answer Follow

answered Nov 25, 2013 at 16:40

peterh
11.6k 18 86 104

1 In C there is no need to cast the results of `malloc/calloc/realloc` nor is it recommended: stackoverflow.com/a/605858/694576 – alk Nov 25, 2013 at 17:40

Until that I did this only to avoid gcc warnings, but his arguments are strong. – peterh Nov 25, 2013 at 23:33

3

When we write

```
char *temp ;
```

it means `temp` is an uninitialized pointer to `char` i.e. currently it does not contain any address in it .

While using `fgets` you have to pass a string in which the bytes read from file pointer is to be copied . link since the `temp` is uninitialized , the `fgets` looks like this

```
fgets(<no string> , 500 , fp ) ;
```

which is invalid .

Hence , we should give initialized string which can be formed as :

```
1) char *temp = malloc(sizeof(500)) ;
or
2) char temp[500] ;
```

Hence if we pass initialized string to `fgets` , it would look like

```
fgets( < some string > , 500 , fp) ;
```

Share Improve this answer Follow

edited Nov 25, 2013 at 18:44

answered Nov 25, 2013 at 16:56

Subbu
2,063 4 29 42

This is a buffer overflow. `malloc(sizeof(500))` will allocate about 4 bytes, not 500. – Bilow Aug 29, 2016 at 14:16

2

`char *temp` is uninitialized, that is, it isn't pointing to valid memory. Either make it an array (`char temp[]`) or use `malloc` to assign memory for it.

Share Improve this answer Follow

answered Nov 25, 2013 at 16:37

Fiddling Bits
8,660 3 28 45