STRING CONCATENATION

strcat() & strncat()

In c there is a function for adding strings. Which is known as **strcat()** functions. This function takes 2 arguments. The first one is destination or in which the strings will be added and the second one is source or from where the strings will be collected from which will be constant.

char *strcat(char *dest, const char *src);

Example:

```c
#include <stdio.h>
#include <string.h>

int main(){
    char str1[100] = "Hello ";
    char str2[100] = "everyone!";
    strcat(str1,str2);
    printf("%s", str1);
}
```

And the output will be `Hello everyone!`

We can also use another variable to store the concated string for which the main variable won't be affected.

```c
#include <stdio.h>
#include <string.h>

int main(){
    char str1[100] = "Hello ";
    char str2[100] = "everyone!";
    char final [100] = "";
    strcat(final,str1);
    strcat(final,str2);
    printf("The string concatation of str1 and str 2 is "
    "%s+ %s = %s",str1,str2,final);
}
```

The output of this one will be

```
The string concatation of str1 and str 2 is Hello + everyone! = Hello everyone!
```

Now, if we want to remove H and only print ‘ello world’ in the output. Then we have to give the source index from where we want to start.

```c
#include <stdio.h>
#include <string.h>

int main(){
    char str1[100] = "Hello ";
    char str2[100] = "everyone!";
    char final [100] = "";
    strcat(final,str1[1]); //here e is at index 1 so we start it from there.
    strcat(final,str2);
    printf("The string concatation of str1 and str 2 is "
    "%s+ %s = %s",str1,str2,final);
}
```

Here, the file will give error because we are taking the character instead of string. To fix this we have to add “&” before the str1[1].

```c
#include <stdio.h>
#include <string.h>

int main(){
    char str1[100] = "Hello ";
    char str2[100] = "everyone!";
    char final [100] = "";
    strcat(final,&str1[1]); //here e is at index 1 so we start it from there.
    strcat(final,str2);
    printf("The string concatation of str1 from the letter e "
    "and str 2 is "
    "%s+ %s = %s",str1,str2,final);
}
```

The output will be

```
The string concatation of str1 from the letter eand str 2 is Hello + everyone! = ello everyone!
```

Note that, if the size of source is larger than the destination size i.e if the size destination is not long enough to accommodate the additional characters of source, then we might face undefined behavior.

But, if we get lucky, it might be possible that the whole string will get stored in destination. But again, this is an undefined behavior, because we don't know what is there beyond the memory allocated to us. Maybe it is possible that there is some critical information stored in the memory beyond the memory allocated to us. So we are calling this is an undefined behavior.

So to avoid this behavior there is another function named **strncat()**. This function have 3 arguments the first two arguments are same as **strcat**. The third argument is an integer which indicates how many character will it concat with the destination from the source.

char *strncat(char *dest, const char *src, n);

n refers to represents a maximum number of characters to be appended.

```c
#include <stdio.h>
#include <string.h>

int main(){
    char str1[100] = "Hello ";
    char str2[40] = "everyone!"; // here we want to take the character upto r
    strncat(str1, str2, 4); // we gave 4 as the argument so that it takes the characters before y
    printf("%s", str1);
}
```

The output of this code will be `Hello ever`

Here the function took the str1 as the destination which contains string “Hello” and str2 as the source from where it will add with the str1 which contains “everyone” now as we want to take the first 4 character of “everyone” we give the last argument 4.

A point to be noted: strncat function automatically adds the null functions at the end of the given arrays.

We can also simply add one simple character from another source string using this function

EXAMPLE

```c
#include <stdio.h>
#include <string.h>

int main(){

    char str1[100] = "strong";
    char final [100] = "";
    //suppose we want to store only s of strong in final
    strncat(final,&str1[0],1);
    printf("%s", final);
}
```

Output = s

This can be useful in some task where we need to work on taking some few characters from any place of a string.

Now it is suggested to use strcat instead of strncat. Because strcat doesn't check the size of additional string data and keep copying until it gets a null character. Whereas strncat check the size of additional string data and concat until n characters.