# Better Tomorrow

## String Manipulations

In C, the standard library provides several string manipulation functions, which are declared in the header file `<string.h>`. Below are descriptions of commonly used string functions:

### 1. `strlen()`

- **Prototype:** `size_t strlen(const char *str);`
- **Description:** Returns the length of the string (excluding the null character `\\0`).
- **Example:**

```
char str[] = "Hello";
size_t len = strlen(str);  // len = 5
```

### 2. `strcpy()`

- **Prototype:** `char *strcpy(char *dest, const char *src);`
- **Description:** Copies the string pointed to by `src` (including the null terminator) to the buffer pointed to by `dest`.
- **Example:**

```
char src[] = "World";
char dest[10];
strcpy(dest, src);  // dest = "World"
```

### 3. `strncpy()`

- **Prototype:** `char *strncpy(char *dest, const char *src, size_t n);`

- **Description:** Copies up to `n` characters from the string pointed to by `src` to the buffer `dest`. If `src` is shorter than `n`, the remainder of `dest` will be padded with null characters.
- **Example:**

```
char src[] = "Hello";
char dest[10];
strncpy(dest, src, 3);  // dest = "Hel"
```

## 4. `strcat()`

- **Prototype:** `char *strcat(char *dest, const char *src);`
- **Description:** Appends the `src` string to the `dest` string (overwriting the null character at the end of `dest`).
- **Example:**

```
char dest[20] = "Hello, ";
char src[] = "World";
strcat(dest, src);  // dest = "Hello, World"
```

## 5. `strncat()`

- **Prototype:** `char *strncat(char *dest, const char *src, size_t n);`
- **Description:** Appends up to `n` characters from `src` to `dest`, ensuring the result is null-terminated.
- **Example:**

```
char dest[20] = "Hello, ";
char src[] = "World";
strncat(dest, src, 3);  // dest = "Hello, Wor"
```

## 6. `strcmp()`

- **Prototype:** `int strcmp(const char *str1, const char *str2);`

- **Description:** Compares two strings lexicographically. Returns 0 if they are equal, a negative value if `str1` is less than `str2`, and a positive value if `str1` is greater than `str2`.
- **Example:**

```
int res = strcmp("abc", "abcd");  // res<0
res = strcmp("abcd", "abc");  // res>0
res = strcmp("abcd", "abcd");  // res=0
```

## 7. `strncmp()`

- **Prototype:** `int strncmp(const char *str1, const char *str2, size_t n);`
- **Description:** Compares up to `n` characters of the two strings.
- **Example:**

```
int res = strncmp("apple", "apricot", 2);  // res = 0
```

## 8. `strchr()`

- **Prototype:** `char *strchr(const char *str, int c);`
- **Description:** Finds the first occurrence of the character `c` in the string `str`. Returns a pointer to the character, or `NULL` if not found.
- **Example:**

```
char *res = strchr("Hello", 'l');  // res points to the fi
rst 'l'
```

## 9. `strrchr()`

- **Prototype:** `char *strrchr(const char *str, int c);`
- **Description:** Finds the last occurrence of the character `c` in the string `str`.
- **Example:**

```
char *res = strrchr("Hello", 'l');  // res points to the l
ast 'l'
```

## 10. `strstr()`

- **Prototype:** `char *strstr(const char *haystack, const char *needle);`
- **Description:** Finds the first occurrence of the substring `needle` in the string `haystack`. Returns a pointer to the beginning of the found substring, or `NULL` if not found.
- **Example:**

```
char *res = strstr("Hello, world", "world");  // res point
s to "world"
```

## 11. `strdup()`

- **Prototype:** `char *strdup(const char *str);`
- **Description:** Returns a pointer to a new string that is a duplicate of `str`. The memory is allocated with `malloc`.
- **Example:**

```
char *copy = strdup("Hello");  // creates a new string "He
llo"
```

## 12. `memset()`

- **Prototype:** `void *memset(void *str, int c, size_t n);`
- **Description:** Fills the first `n` bytes of the memory area pointed to by `str` with the constant byte `c`.
- **Example:**

```
char buffer[10];
```

```
memset(buffer, 0, 10);  // fills buffer with zeros
```

### 13. `memcpy()`

- **Prototype:** `void *memcpy(void *dest, const void *src, size_t n);`
- **Description:** Copies `n` bytes from the memory area `src` to `dest`.
- **Example:**

```
char src[] = "Data";
char dest[10];
memcpy(dest, src, 5);  // copies "Data" to dest
```

### 14. `memmove()`

- **Prototype:** `void *memmove(void *dest, const void *src, size_t n);`
- **Description:** Similar to `memcpy`, but handles overlapping memory areas.
- **Example:**

```
char str[] = "Overlap";
memmove(str + 2, str, 5);  // shifts "Overl" two positions
forward
```

## Example Code

```
#include<stdio.h>
#include<string.h>
int main()
{
    char a[]="apple";
    char b[]="application";
    char c[20];
    printf("%ld ",strlen(a));//5
    strcpy(c,a);//c="apple"
    printf("%s ",c);
    strncpy(c,b,5);//c="appli"
    printf("%s ",c);
    strcat(c,a);//c="appliapple"
    printf("%s ",c);
    strncat(c,b,3);//c="appliappleapp"
    printf("%s ",c);
    printf("%d ",strcmp(a,b));//-4
    printf("%d ",strncmp(a,b,4));//0
    printf("%s ",strchr(a,'p'));//pple
    printf("%s ",strrchr(a,'p'));//ple
    printf("%s ",strstr(b,"cat"));//cation
    char *d=strdup(b);//d="application"
    printf("%s\n",d);
    return 0;
}
```

## Task

💡 By understanding above in-built functions, write custom functions to do perform similar functionalities **without using library functions.**

- strlen
- strcpy
- strncpy
- strcmp
- strncmp
- strcat
- strncat
- strchr

- strrchr
- strstr
- strdup

Better Tomorrow - Training Institute

Better Tomorrow Training Institute - Product based training, MERN stack, Programming, Learning, Placement Training

http://www.thebettertomorrow.in

Follow us on our : LinkedIn Page