**NAME**

**strcmp** – compare two strings

**SYNOPSIS**

```
#include <string.h>

int strcmp(const char *s1, const char *s2);
```

**DESCROPTION**

The **strcmp()** function compares the ASCII value of the two strings s1 and s2. It returns 0 if s1 and s2 are the same; returns a negative integer if s1 is smaller than s2; returns a positive integer if s1 is larger than s2.

**RETURN VALUE**

The **strcmp()** function returns 0 if s1 matches s2; it returns a negative integer if the unmatched character in s1 is less than the unmatched character in s2; it returns a positive integer if the unmatched character in s1 is larger than the unmatched character in s2.

**ERROR**

First: s1 or s2 are not found;
Second: the argument is not a string;

**SOURCE CODE**

```
/*Firstly, set a return value: rvalue, if s1 and s2
  are not the same; because strcmp() return the
  difference of ASCII values of the first unmatched
  character of two strings;
  Then, set a counter i to traverse the string; After
  that, we can start compare each character in two
  strings to find when does the first unmatched
  character appear;
  if two strings are the same, we need to reach to
  the end of the string, and string terminator is
  needed;
  if s1 is smaller or larger than s2, strcmp returns
  s1[i] - s2[i];
  if s1 == s2, strcmp returns 0;
*/
```

```
int mystrcmp(const char *s1, const char *s2) {
  int rvalue;
  int i = 0;
  while ((s1[i] == s2[i]) && (s1[i] != '\0' &&
  s2[i] != '\0')){
    i++;
  }
  if (s1[i] != s2[i]){
    rvalue = (s1[i] - s2[i]);
    return rvalue;
  }
  else
  return 0;
}
```

**NAME**

**Strdup –** duplicate a string;

**SYNOPSIS**

```
#include <string.h>

char *strdup(const char *s);
```

**DESCRIPTION**

The **strdup()** function duplicates the string s to a new string by obtaining the memory for the new string using malloc(); The new string has the same content as string s, but the memory address is different. The memory for the new string can be freed with free().

**RETURN VALUE**

the **strdup**() function returns a pointer to the new duplicated string which has the same content as string s. It returns NULL if insufficient memory was available, with errno set to indicate the cause of the error.

**ERRORS**

First: ENOMEM Insufficient memory available to allocate duplicate string.
Second: the argument is not a string;

**SOURCE CODE**

```
/*If s is NULL, return NULL;
Create a counter j to count the number of elements
in the string s; Then give a new memory address to
store the new duplicate string;
After that, traverse the string s to copy the content
into the new duplicate string;
And the duplicate string ends with '\0';
*/

char *mystrdup(const char *s) {
  if (s == NULL){
    return NULL;
  }
  int j = 0;
  while(s[j] != '\0'){
```

```
    j++;
  }
  char *new;
  new = malloc(j);
  int k = 0;
  while(s[k] != '\0'){
    new[k] = s[k];
    k++;
  }
  new[k] = '\0';
  return new;
}
```

2019-10-30. STRDUP()