< [A Little C Primer](https://en.wikibooks.org/wiki/A_Little_C_Primer)

[**A Little C Primer**](https://en.wikibooks.org/wiki/A_Little_C_Primer)

The string-function library requires the declaration:

#include *<string.h>*

The most important string functions are as follows:

strlen() Get length of a string.

strcpy() Copy one string to another.

strcat() Link together (concatenate) two strings.

strcmp() Compare two strings.

strchr() Find character in string.

strstr() Find string in string.

strlwr() Convert string to lowercase.

strupr() Convert string to uppercase.

**Contents**

  [hide]

* [1strlen()](https://en.wikibooks.org/wiki/A_Little_C_Primer/C_String_Function_Library#strlen.28.29)
* [2strcpy()](https://en.wikibooks.org/wiki/A_Little_C_Primer/C_String_Function_Library#strcpy.28.29)
* [3strncpy()](https://en.wikibooks.org/wiki/A_Little_C_Primer/C_String_Function_Library#strncpy.28.29)
* [4strcat()](https://en.wikibooks.org/wiki/A_Little_C_Primer/C_String_Function_Library#strcat.28.29)
* [5strncat()](https://en.wikibooks.org/wiki/A_Little_C_Primer/C_String_Function_Library#strncat.28.29)
* [6strcmp()](https://en.wikibooks.org/wiki/A_Little_C_Primer/C_String_Function_Library#strcmp.28.29)
  + [6.1strncmp()](https://en.wikibooks.org/wiki/A_Little_C_Primer/C_String_Function_Library#strncmp.28.29)
  + [6.2stricmp()](https://en.wikibooks.org/wiki/A_Little_C_Primer/C_String_Function_Library#stricmp.28.29)
  + [6.3strnicmp()](https://en.wikibooks.org/wiki/A_Little_C_Primer/C_String_Function_Library#strnicmp.28.29)
* [7strchr()](https://en.wikibooks.org/wiki/A_Little_C_Primer/C_String_Function_Library#strchr.28.29)
  + [7.1strrchr()](https://en.wikibooks.org/wiki/A_Little_C_Primer/C_String_Function_Library#strrchr.28.29)
* [8strstr()](https://en.wikibooks.org/wiki/A_Little_C_Primer/C_String_Function_Library#strstr.28.29)
* [9strlwr() and strupr()](https://en.wikibooks.org/wiki/A_Little_C_Primer/C_String_Function_Library#strlwr.28.29_and_strupr.28.29)
* [10further reading](https://en.wikibooks.org/wiki/A_Little_C_Primer/C_String_Function_Library#further_reading)

**strlen()**[[edit](https://en.wikibooks.org/w/index.php?title=A_Little_C_Primer/C_String_Function_Library&action=edit&section=1)]

The "strlen()" function gives the length of a string, not including the NUL character at the end:

*/\* strlen.c \*/*

#include *<stdio.h>*

#include *<string.h>*

int main()

{

char \*t = "XXX";

printf( "Length of <%s> is %d.**\n**", t, strlen( t ));

}

This prints:

Length of <XXX> is 3.

**strcpy()**[[edit](https://en.wikibooks.org/w/index.php?title=A_Little_C_Primer/C_String_Function_Library&action=edit&section=2)]

The "strcpy" function copies one string from another. For example:

*/\* strcpy.c \*/*

#include *<stdio.h>*

#include *<string.h>*

int main()

{

char s1[100],

s2[100];

strcpy( s1, "xxxxxx 1" );

strcpy( s2, "zzzzzz 2" );

puts( "Original strings: " );

puts( "" );

puts( s1 );

puts( s2 );

puts( "" );

strcpy( s2, s1 );

puts( "New strings: " );

puts( "" );

puts( s1 );

puts( s2 );

}

This will print:

Original strings:

xxxxxx 1

zzzzzz 2

New strings:

xxxxxx 1

xxxxxx 1

Please be aware of two features of this program:

* This program assumes that "s1" has enough space to store the final string. The "strcpy()" function won't bother to check, and will give erroneous results if that is not the case.
* A string constant can be used as the source string instead of a string variable. Using a string constant for the destination, of course, makes no sense.

These comments are applicable to most of the other string functions.

**strncpy()**[[edit](https://en.wikibooks.org/w/index.php?title=A_Little_C_Primer/C_String_Function_Library&action=edit&section=3)]

There is a variant form of "strcpy" named "strncpy" that will copy "n" characters of the source string to the destination string, presuming there are that many characters available in the source string. For example, if the following change is made in the example program:

strncpy( s2, s1, 5 );

-- then the results change to:

New strings:

xxxxxx 1

xxxxxz 2

Notice that the parameter "n" is declared "size\_t", which is defined in "string.h". Because strncpy does not add '\0' after coping 5 charecters

**strcat()**[[edit](https://en.wikibooks.org/w/index.php?title=A_Little_C_Primer/C_String_Function_Library&action=edit&section=4)]

The "strcat()" function joins two strings:

*/\* strcat.c \*/*

#include *<stdio.h>*

#include *<string.h>*

int main()

{

char s1[50],

s2[50];

strcpy( s1, "Tweedledee " );

strcpy( s2, "Tweedledum" );

strcat( s1, s2 );

puts( s1 );

}

This prints:

Tweedledee Tweedledum

**strncat()**[[edit](https://en.wikibooks.org/w/index.php?title=A_Little_C_Primer/C_String_Function_Library&action=edit&section=5)]

There is a variant version of "strcat()" named "strncat()" that will append "n" characters of the source string to the destination string. If the example above used "strncat()" with a length of 7:

strncat( s1, s2, 7 );

-- the result would be:

Tweedledee Tweedle

Again, the length parameter is of type "size\_t".

**strcmp()**[[edit](https://en.wikibooks.org/w/index.php?title=A_Little_C_Primer/C_String_Function_Library&action=edit&section=6)]

The "strcmp()" function compares two strings:

*/\* strcmp.c \*/*

#include *<stdio.h>*

#include *<string.h>*

#define ANSWER "blue"

int main()

{

char t[100];

puts( "What is the secret color?" );

gets( t );

**while** ( strcmp( t, ANSWER ) != 0 )

{

puts( "Wrong, try again." );

gets( t );

}

puts( "Right!" );

}

The "strcmp()" function returns a 0 for a successful comparison, and nonzero otherwise. The comparison is case-sensitive, so answering "BLUE" or "Blue" won't work.

There are three alternate forms for "strcmp()":

**strncmp()**[[edit](https://en.wikibooks.org/w/index.php?title=A_Little_C_Primer/C_String_Function_Library&action=edit&section=7)]

* A "strncmp()" function which, as might be guessed, compares "n" characters in the source string with the destination string:

"strncmp( s1, s2, 6 )".

**stricmp()**[[edit](https://en.wikibooks.org/w/index.php?title=A_Little_C_Primer/C_String_Function_Library&action=edit&section=8)]

* A "stricmp()" function that ignores case in comparisons.

**strnicmp()**[[edit](https://en.wikibooks.org/w/index.php?title=A_Little_C_Primer/C_String_Function_Library&action=edit&section=9)]

* A case-insensitive version of "strncmp" called "strnicmp".

**strchr()**[[edit](https://en.wikibooks.org/w/index.php?title=A_Little_C_Primer/C_String_Function_Library&action=edit&section=10)]

The "strchr" function finds the first occurrence of a character in a string. It returns a pointer to the character if it finds it, and null if not. For example:

*/\* strchr.c \*/*

#include *<stdio.h>*

#include *<string.h>*

int main()

{

char \*t = "MEAS:VOLT:DC?";

char \*p;

p = t;

puts( p );

**while**(( p = strchr( p, ':' )) != NULL )

{

puts( ++p );

}

}

This prints:

MEAS:VOLT:DC?

VOLT:DC?

DC?

The character is defined as a character constant, which C regards as an "int". Notice how the example program increments the pointer before using it ("++p") so that it doesn't point to the ":" but to the character following it.

**strrchr()**[[edit](https://en.wikibooks.org/w/index.php?title=A_Little_C_Primer/C_String_Function_Library&action=edit&section=11)]

The "strrchr()" function is almost the same as "strchr()", except that it searches for the *last* occurrence of the character in the string.

**strstr()**[[edit](https://en.wikibooks.org/w/index.php?title=A_Little_C_Primer/C_String_Function_Library&action=edit&section=12)]

The "strstr()" function is similar to "strchr()" except that it searches for a string, instead of a character. It also returns a pointer:

char \*s = "Black White Brown Blue Green";

...

puts( strstr( s, "Blue" ) );

**strlwr() and strupr()**[[edit](https://en.wikibooks.org/w/index.php?title=A_Little_C_Primer/C_String_Function_Library&action=edit&section=13)]

The "strlwr()" and "strupr()" functions simply perform lowercase or uppercase conversion on the source string. For example:

*/\* casecvt.c \*/*

#include *<stdio.h>*

#include *<string.h>*

int main()

{

char \*t = "Hey Barney hey!";

puts( strlwr( t ) );

puts( strupr( t ) );

}

-- prints:

hey barney hey!

HEY BARNEY HEY!

These two functions are only implemented in some compilers and are not part of ANSI C.