strcpy, strcpy_s

Defined in header <string.h>

`char *strcpy( char *dest, const char *src );`	(1)	(until C99)
`char *strcpy( char *restrict dest, const char *restrict src );`		(since C99)
`errno_t strcpy_s(char *restrict dest, rsize_t destsz, const char *restrict src);`	(2)	(since C11)

1) Copies the null-terminated byte string pointed to by `src`, including the null terminator, to the character array whose first element is pointed to by `dest`.

The behavior is undefined if the `dest` array is not large enough. The behavior is undefined if the strings overlap. The behavior is undefined if either `dest` is not a pointer to a character array or `src` is not a pointer to a null-terminated byte string.

2) Same as (1), except that it may clobber the rest of the destination array with unspecified values and that the following errors are detected at runtime and call the currently installed constraint handler function:

- `src` or `dest` is a null pointer
- `destsz` is zero or greater than `RSIZE_MAX`
- `destsz` is less or equal `strnlen_s(src, destsz)`; in other words, truncation would occur
- overlap would occur between the source and the destination strings

The behavior is undefined if the size of the character array pointed to by `dest` <= `strnlen_s(src, destsz)` < `destsz`; in other words, an erroneous value of `destsz` does not expose the impending buffer overflow.

As with all bounds-checked functions, `strcpy_s` is only guaranteed to be available if `__STDC_LIB_EXT1__` is defined by the implementation and if the user defines `__STDC_WANT_LIB_EXT1__` to the integer constant 1 before including `string.h`.

Parameters

dest	-	pointer to the character array to write to
src	-	pointer to the null-terminated byte string to copy from
destsz	-	maximum number of characters to write, typically the size of the destination buffer

Return value

1) returns a copy of `dest`

2) returns zero on success, returns non-zero on error. Also, on error, writes zero to `dest[0]` (unless `dest` is a null pointer or `destsz` is zero or greater than `RSIZE_MAX`).

Notes

`strcpy_s` is allowed to clobber the destination array from the last character written up to `destsz` in order to improve efficiency: it may copy in multibyte blocks and then check for null bytes.

The function `strcpy_s` is similar to the BSD function `strlcpy`, except that

- `strlcpy` truncates the source string to fit in the destination (which is a security risk)
- `strlcpy` does not perform all the runtime checks that `strcpy_s` does
- `strlcpy` does not make failures obvious by setting the destination to a null string or calling a handler if the call fails.

Although `strcpy_s` prohibits truncation due to potential security risks, it's possible to truncate a string using bounds-checked `strncpy_s` instead.

Example

Run this code

```
#define __STDC_WANT_LIB_EXT1__ 1
#include <string.h>
#include <stdio.h>
```

```c
#include <stdlib.h>

int main(void)
{
    char *src = "Take the test.";
//  src[0] = 'M' ; // this would be undefined behavior
    char dst[strlen(src) + 1]; // +1 to accomodate for the null terminator
    strcpy(dst, src);
    dst[0] = 'M'; // OK
    printf("src = %s\ndst = %s\n", src, dst);

#ifdef __STDC_LIB_EXT1__
    set_constraint_handler_s(ignore_handler_s);
    int r = strcpy_s(dst, sizeof dst, src);
    printf("dst = \"%s\", r = %d\n", dst, r);
    r = strcpy_s(dst, sizeof dst, "Take even more tests.");
    printf("dst = \"%s\", r = %d\n", dst, r);
#endif
}
```

Possible output:

```
src = Take the test.
dst = Make the test.
dst = "Take the test.", r = 0
dst = "", r = 22
```

References

- C11 standard (ISO/IEC 9899:2011):
  - 7.24.2.3 The strcpy function (p: 363)
  - K.3.7.1.3 The strcpy_s function (p: 615-616)
- C99 standard (ISO/IEC 9899:1999):
  - 7.21.2.3 The strcpy function (p: 326)
- C89/C90 standard (ISO/IEC 9899:1990):
  - 4.11.2.3 The strcpy function

See also

strncpy **strncpy_s** (C11)	copies a certain amount of characters from one string to another (function)
memcpy **memcpy_s** (C11)	copies one buffer to another (function)
wcscpy (C95) **wcscpy_s** (C11)	copies one wide string to another (function)
strdup (dynamic memory TR)	allocate a copy of a string (function)

C++ documentation for **strcpy**