

**NAME**

**stpncpy**, **stpcpy**, **strcpy**, **strncpy** - copy strings

**LIBRARY**

Standard C Library (libc, -lc)

**SYNOPSIS**

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

```
char *
```

```
strcpy(char * restrict dst, const char * restrict src);
```

```
char *
```

```
stpcpy(char * restrict dst, const char * restrict src, size_t len);
```

```
char *
```

```
strcpy(char * restrict dst, const char * restrict src);
```

```
char *
```

```
strncpy(char * restrict dst, const char * restrict src, size_t len);
```

**DESCRIPTION**

The **strcpy()** and **stpcpy()** functions copy the string *src* to *dst* (including the terminating ‘\0’ character.)

The **strncpy()** and **stpcpy()** functions copy at most *len* characters from *src* into *dst*. **If *src* is less than *len* characters long, the remainder of *dst* is filled with ‘\0’ characters.** Otherwise, *dst* is *not* terminated.

For all of **strcpy()**, **strncpy()**, **stpcpy()**, and **stpcpy()**, the result is undefined if *src* and *dst* overlap.

**RETURN VALUES**

The **strcpy()** and **strncpy()** functions return *dst*. The **stpcpy()** and **stpcpy()** functions return a pointer to the terminating ‘\0’ character of *dst*. If **stpcpy()** does not terminate *dst* with a NUL character, it instead returns a pointer to *dst*[*n*] (which does not necessarily refer to a valid memory location.)

**EXAMPLES**

The following sets *chararray* to "abc\0\0\0":

```
char chararray[6];
```

```
(void)strncpy(chararray, "abc", sizeof(chararray));
```

The following sets *chararray* to "abcdef":

```
char chararray[6];

(void)strncpy(chararray, "abcdefgh", sizeof(chararray));
```

Note that it does *not* NUL terminate *chararray* because the length of the source string is greater than or equal to the length argument.

The following copies as many characters from *input* to *buf* as will fit and NUL terminates the result. Because **strncpy()** does *not* guarantee to NUL terminate the string itself, this must be done explicitly.

```
char buf[1024];

(void)strncpy(buf, input, sizeof(buf) - 1);
buf[sizeof(buf) - 1] = '\0';
```

This could be better achieved using **strncpy(3)**, as shown in the following example:

```
(void)strncpy(buf, input, sizeof(buf));
```

## SEE ALSO

**bcopy(3)**, **memcpy(3)**, **memmove(3)**, **strncpy(3)**, **wscpy(3)**

## STANDARDS

The **strcpy()** and **strncpy()** functions conform to ISO/IEC 9899:1990 ("ISO C90"). The **strcpy()** and **strncpy()** functions conform to IEEE Std 1003.1-2008 ("POSIX.1").

## HISTORY

The **strcpy()** function first appeared in FreeBSD 4.4, and **strncpy()** was added in FreeBSD 8.0.

## SECURITY CONSIDERATIONS

All of the functions documented in this manual page are easily misused in a manner which enables malicious users to arbitrarily change a running program's functionality through a buffer overflow attack.

It is strongly suggested that the **strcpy()** function be used in almost all cases.

For some, but not all, fixed-length records, non-terminated strings may be both valid and desirable. In that specific case, the **strncpy()** function may be most sensible.