NAME

strncat − concatenate a null-padded character sequence into a string

LIBRARY

Standard C library (*libc*, *−lc*)

SYNOPSIS

#include <string.h>

char *strncat(char *restrict *dst***, const char** *src***[restrict .***sz***],**
size_t *sz***);**

DESCRIPTION

This function catenates the input character sequence contained in a null-padded fixed-width buffer, into a string at the buffer pointed to by *dst*. The programmer is responsible for allocating a destination buffer large enough, that is, *strlen(dst) + strnlen(src, sz) + 1*.

An implementation of this function might be:

```
char *
strncat(char *restrict dst, const char *restrict src, size_t sz)
{
    int   len;
    char  *p;

    len = strnlen(src, sz);
    p = dst + strlen(dst);
    p = mempcpy(p, src, len);
    *p = '\0';

    return dst;
}
```

RETURN VALUE

strncat() returns *dst*.

ATTRIBUTES

For an explanation of the terms used in this section, see **attributes**(7).

Interface	Attribute	Value
strncat()	Thread safety	MT-Safe

STANDARDS

POSIX.1-2001, POSIX.1-2008, C99, SVr4, 4.3BSD.

CAVEATS

The name of this function is confusing. This function has no relation to **strncpy**(3).

If the destination buffer is not large enough, the behavior is undefined. See **_FORTIFY_SOURCE** in **feature_test_macros**(7).

BUGS

This function can be very inefficient. Read about Shlemiel thepainter⟨https://www .joelonsoftware.com/2001/12/11/back−to−basics/⟩.

EXAMPLES

```
#include <err.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define nitems(arr)  (sizeof((arr)) / sizeof((arr)[0]))
```

```
int
main(void)
{
    size_t  maxsize;

    // Null-padded fixed-width character sequences
    char    pre[4] = "pre.";
    char    new_post[50] = ".foo.bar";

    // Strings
    char    post[] = ".post";
    char    src[] = "some_long_body.post";
    char    *dest;

    maxsize = nitems(pre) + strlen(src) − strlen(post) +
              nitems(new_post) + 1;
    dest = malloc(sizeof(*dest) * maxsize);
    if (dest == NULL)
        err(EXIT_FAILURE, "malloc()");

    dest[0] = '\0';  // There’s no ’cpy’ function to this ’cat’.
    strncat(dest, pre, nitems(pre));
    strncat(dest, src, strlen(src) − strlen(post));
    strncat(dest, new_post, nitems(new_post));

    puts(dest);  // "pre.some_long_body.foo.bar"
    free(dest);
    exit(EXIT_SUCCESS);
}
```

SEE ALSO

string(3), **string_copying**(3)