NAME

strcat, strncat – concatenate two strings

SYNOPSIS

#include <string.h>

char *strcat(char **dest*, const char **src*);

char *strncat(char **dest*, const char **src*, size_t *n*);

DESCRIPTION

The **strcat**() function appends the *src* string to the *dest* string, overwriting the terminating null byte ('\0') at the end of *dest*, and then adds a terminating null byte. The strings may not overlap, and the *dest* string must have enough space for the result. If *dest* is not large enough, program behavior is unpredictable; *buffer overruns are a favorite avenue for attacking secure programs*.

The **strncat**() function is similar, except that

* it will use at most *n* bytes from *src*; and
* *src* does not need to be null-terminated if it contains *n* or more bytes.

As with **strcat**(), the resulting string in *dest* is always null-terminated.

If *src* contains *n* or more bytes, **strncat**() writes *n+1* bytes to *dest* (*n* from *src* plus the terminating null byte). Therefore, the size of *dest* must be at least *strlen(dest)+n+1*.

A simple implementation of **strncat**() might be:

```
char *
strncat(char *dest, const char *src, size_t n)
{
    size_t dest_len = strlen(dest);
    size_t i;

    for (i = 0 ; i < n && src[i] != '\0' ; i++)
        dest[dest_len + i] = src[i];
    dest[dest_len + i] = '\0';

    return dest;
}
```

RETURN VALUE

The **strcat**() and **strncat**() functions return a pointer to the resulting string *dest*.

ATTRIBUTES

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

Interface	Attribute	Value
strcat(), **strncat**()	Thread safety	MT-Safe

CONFORMING TO

POSIX.1-2001, POSIX.1-2008, C89, C99, SVr4, 4.3BSD.

NOTES

Some systems (the BSDs, Solaris, and others) provide the following function:

size_t strlcat(char *dest, const char *src, size_t size);

This function appends the null-terminated string *src* to the string *dest*, copying at most *size−strlen(dest)−1* from *src*, and adds a terminating null byte to the result, *unless size* is less than *strlen(dest)*. This function fixes the buffer overrun problem of **strcat**(), but the caller must still handle the possibility of data loss if *size* is too small. The function returns the length of the string **strlcat**() tried to create; if the return value is greater than or equal to *size*, data loss occurred. If data loss matters, the caller *must* either check the arguments before the call, or test the function return value. **strlcat**() is not present in glibc and is not

standardized by POSIX, but is available on Linux via the *libbsd* library.

EXAMPLE

Because **strcat**() and **strncat**() must find the null byte that terminates the string *dest* using a search that starts at the beginning of the string, the execution time of these functions scales according to the length of the string *dest*. This can be demonstrated by running the program below. (If the goal is to concatenate many strings to one target, then manually copying the bytes from each source string while maintaining a pointer to the end of the target string will provide better performance.)

Program source

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

int
main(int argc, char *argv[])
{
#define LIM 4000000
    int j;
    char p[LIM + 1];    /* +1 for terminating null byte */
    time_t base;

    base = time(NULL);
    p[0] = '\0';

    for (j = 0; j < LIM; j++) {
        if ((j % 10000) == 0)
            printf("%d %ld\n", j, (long) (time(NULL) - base));
        strcat(p, "a");
    }
}
```

SEE ALSO

bcopy(3), **memccpy**(3), **memcpy**(3), **strcpy**(3), **string**(3), **strncpy**(3), **wcscat**(3), **wcsncat**(3)

COLOPHON

This page is part of release 5.02 of the Linux *man-pages* project. A description of the project, information about reporting bugs, and the latest version of this page, can be found at https://www.kernel.org/doc/man−pages/.