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

Multithreading (see pthreads(7))

The **strcat**() and **strncat**() functions are thread-safe.

CONFORMING TO

SVr4, 4.3BSD, C89, C99.

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.

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 3.74 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 http://www.kernel.org/doc/man−pages/.