NAME

strtoul, strtoull, strtouq − convert a string to an unsigned long integer

LIBRARY

Standard C library (*libc*, *−lc*)

SYNOPSIS

#include <stdlib.h>

unsigned long strtoul(const char *restrict *nptr***,**
char **restrict *endptr***, int** *base***);**
unsigned long long strtoull(const char *restrict *nptr***,**
char **restrict *endptr***, int** *base***);**

Feature Test Macro Requirements for glibc (see **feature_test_macros**(7)):

strtoull():
_ISOC99_SOURCE
|| /* glibc <= 2.19: */ _SVID_SOURCE || _BSD_SOURCE

DESCRIPTION

The **strtoul**() function converts the initial part of the string in *nptr* to an *unsigned long* value according to the given *base*, which must be between 2 and 36 inclusive, or be the special value 0.

The string may begin with an arbitrary amount of white space (as determined by **isspace**(3)) followed by a single optional '+' or '−' sign. If *base* is zero or 16, the string may then include a "0x" prefix, and the number will be read in base 16; otherwise, a zero *base* is taken as 10 (decimal) unless the next character is '0', in which case it is taken as 8 (octal).

The remainder of the string is converted to an *unsigned long* value in the obvious manner, stopping at the first character which is not a valid digit in the given base. (In bases above 10, the letter 'A' in either uppercase or lowercase represents 10, 'B' represents 11, and so forth, with 'Z' representing 35.)

If *endptr* is not NULL, **strtoul**() stores the address of the first invalid character in**endptr*. If there were no digits at all, **strtoul**() stores the original value of *nptr* in **endptr* (and returns 0). In particular, if**nptr* is not '\0' but ***endptr* is '\0' on return, the entire string is valid.

The **strtoull**() function works just like the **strtoul**() function but returns an *unsigned long long* value.

RETURN VALUE

The **strtoul**() function returns either the result of the conversion or, if there was a leading minus sign, the negation of the result of the conversion represented as an unsigned value, unless the original (nonnegated) value would overflow; in the latter case, **strtoul**() returns **ULONG_MAX** and sets *errno* to **ERANGE**. Precisely the same holds for **strtoull**() (with **ULLONG_MAX** instead of **ULONG_MAX**).

ERRORS

EINVAL
(not in C99) The given *base* contains an unsupported value.

ERANGE
The resulting value was out of range.

The implementation may also set *errno* to **EINVAL** in case no conversion was performed (no digits seen, and 0 returned).

ATTRIBUTES

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

Interface	**Attribute**	**Value**
strtoul(), **strtoull**(), **strtouq**()	Thread safety	MT-Safe locale

STANDARDS

strtoul(): POSIX.1-2001, POSIX.1-2008, C99, SVr4.

strtoull(): POSIX.1-2001, POSIX.1-2008, C99.

NOTES

Since **strtoul**() can legitimately return 0 or **ULONG_MAX** (**ULLONG_MAX** for **strtoull**()) on both success and failure, the calling program should set *errno* to 0 before the call, and then determine if an error occurred by checking whether *errno* has a nonzero value after the call.

In locales other than the "C" locale, other strings may be accepted. (For example, the thousands separator of the current locale may be supported.)

BSD also has

```
u_quad_t strtouq(const char *nptr, char **endptr, int base);
```

with completely analogous definition. Depending on the wordsize of the current architecture, this may be equivalent to **strtoull**() or to **strtoul**().

Negative values are considered valid input and are silently converted to the equivalent *unsigned long* value.

EXAMPLES

See the example on the **strtol**(3) manual page; the use of the functions described in this manual page is similar.

SEE ALSO

a64l(3), **atof**(3), **atoi**(3), **atol**(3), **strtod**(3), **strtol**(3), **strtoumax**(3)