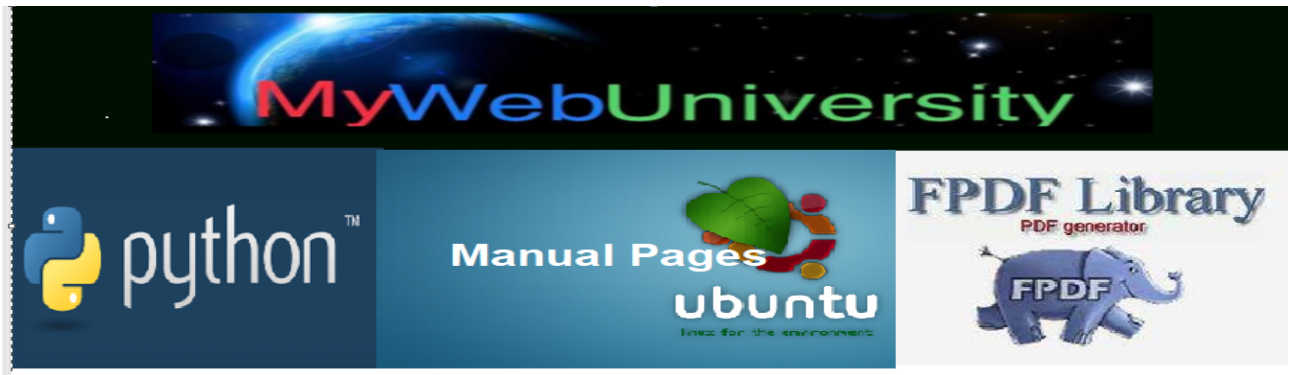

Full credit is given to the above companies including the OS that this PDF file was generated!

Linux Ubuntu 26.04 Manual Pages on command 'strtol.3'

$ man strtol.3

strtol(3) Library Functions Manual strtol(3)

NAME

strtol, strtoll, strtoq - convert a string to a long integer

LIBRARY

Standard C library (libc, -lc)

SYNOPSIS

```
#include <stdlib.h>

long strtol(const char *restrict nptr,
            char **_Nullable restrict endptr, int base);

long long strtoll(const char *restrict nptr,
            char **_Nullable restrict endptr, int base);
```

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

strtoll():

_ISOC99_SOURCE

|| /* glibc <= 2.19: */ _SVID_SOURCE || _BSD_SOURCE

DESCRIPTION

The strtol() function converts the initial part of the string in nptr to a long integer value according to the given base, which must be between 2 and 36 inclusive, or be the spe? cial 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" or "0X" prefix, and the number will be read in base 16; if base is zero or 2, the string may then include a "0b" or "0B" prefix, and the number will be read in base 2;

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 a 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, and the base is supported, strtol() stores the address of the first invalid character in *endptr. If there were no digits at all, strtol() 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 strtoll() function works just like the strtol() function but returns a long long integer value.

RETURN VALUE

The strtol() function returns the result of the conversion, unless the value would underflow or overflow. If an underflow occurs, strtol() returns LONG_MIN. If an overflow occurs, strtol() returns LONG_MAX. In both cases, errno is set to ERANGE. Precisely the same holds for strtoll() (with LLONG_MIN and LLONG_MAX instead of LONG_MIN and LONG_MAX).

ERRORS

This function does not modify errno on success.

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 dig? its seen, and 0 returned).

ATTRIBUTES

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

Interface	Attribute	Value
strtol(), strtoll(), strtoq()	Thread safety	MT-Safe locale

VERSIONS

According to POSIX.1, in locales other than "C" and "POSIX", these functions may accept other, implementation-defined numeric strings.

BSD also has

```
quad_t strtoq(const char *nptr, char **endptr, int base);
```

with completely analogous definition. Depending on the wordsize of the current architec? ture, this may be equivalent to strtoll() or to strtol().

STANDARDS

C23, POSIX.1-2024.

HISTORY

strtol()

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

strtoll()

POSIX.1-2001, C99.

"0b", "0B"

C23. glibc 2.38. (Not in POSIX.)

CAVEATS

Range checks

Since strtol() can legitimately return 0, LONG_MAX, or LONG_MIN (LLONG_MAX or LLONG_MIN for strtoll()) 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 == ERANGE after the call.

```
errno = 0;
n = strtol(s, &end, base);
if (end == s)
    goto no_number;
if ((errno == ERANGE && n == _Minof(long)) || n < min)
    goto too_low;
if ((errno == ERANGE && n == _Maxof(long)) || n > max)
    goto too_high;
```

base

If the base needs to be tested, it should be tested in a call where the string is known to succeed. Otherwise, it's impossible to portably differentiate the errors.

```
errno = 0;
strtol("0", NULL, base);
if (errno == EINVAL)
```

```
        goto unsupported_base;
```

BUGS

White space

These functions silently accept leading white space. To reject white space, call isspace(3) before strtol().

EXAMPLES

The program shown below demonstrates the use of strtol(). The first command-line argument specifies a string from which strtol() should parse a number. The second (optional) argument specifies the base to be used for the conversion. (This argument is converted to numeric form using atoi(3), a function that performs no error checking and has a simpler interface than strtol().) Some examples of the results produced by this program are the following:

```
    $ ./a.out 123
    strtol() returned 123
    $ ./a.out '    123'
    strtol() returned 123
    $ ./a.out 123abc
    strtol() returned 123
    Further characters after number: "abc"
    $ ./a.out 123abc 55
    strtol: Invalid argument
    $ ./a.out ''
    No digits were found
    $ ./a.out 4000000000
    strtol: Numerical result out of range
```

Program source

```
#include <errno.h>
#include <stdio.h>
#include <stdlib.h>
int
main(int argc, char *argv[])
{
    int base;
```

```
    char *endptr, *str;
    long val;
    if (argc < 2) {
        fprintf(stderr, "Usage: %s str [base]\n", argv[0]);
        exit(EXIT_FAILURE);
    }
    str = argv[1];
    base = (argc > 2) ? atoi(argv[2]) : 0;
    errno = 0;    /* To distinguish success/failure after call */
    strtol("0", NULL, base);
    if (errno == EINVAL) {
        perror("strtol");
        exit(EXIT_FAILURE);
    }
    errno = 0;    /* To distinguish success/failure after call */
    val = strtol(str, &endptr, base);
    /* Check for various possible errors.  */
    if (errno == ERANGE) {
        perror("strtol");
        exit(EXIT_FAILURE);
    }
    if (endptr == str) {
        fprintf(stderr, "No digits were found\n");
        exit(EXIT_FAILURE);
    }
    /* If we got here, strtol() successfully parsed a number.  */
    printf("strtol() returned %ld\n", val);
    if (*endptr != '\0')        /* Not necessarily an error.  */
        printf("Further characters after number: \"%s\"\n", endptr);
    exit(EXIT_SUCCESS);
}
```

SEE ALSO

atof(3), atoi(3), atol(3), strtod(3), strtoimax(3), strtoul(3)