



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

Rocky Enterprise Linux 9.2 Manual Pages on command 'strtol.3'

\$ man strtol.3

STRTOL(3) Linux Programmer's Manual STRTOL(3)

NAME

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

SYNOPSIS

```
#include <stdlib.h>
```

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

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

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

```
strtoll():
```

```
  _ISOC99_SOURCE
```

```
  || /* Glibc versions <= 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 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" or "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 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

other, implementation-defined numeric strings.

BSD also has

```
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 strtoll() or to 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 <stdlib.h>
```

```
#include <limits.h>
```

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

```
#include <errno.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 */

val = strtol(str, &endptr, base);

/* Check for various possible errors */

if (errno != 0) {
    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), strtointmax(3), strtoul(3),

COLOPHON

This page is part of release 5.10 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/>.