strtol, strtoll

Defined in header <stdlib.h>

```
long      strtol( const char          *str, char          **str_end, int base );      (until C99)
long      strtol( const char *restrict str, char **restrict str_end, int base );      (since C99)
long long strtoll( const char *restrict str, char **restrict str_end, int base );    (since C99)
```

Interprets an integer value in a byte string pointed to by `str`.

Discards any whitespace characters (as identified by calling `isspace`) until the first non-whitespace character is found, then takes as many characters as possible to form a valid *base-n* (where n=`base`) integer number representation and converts them to an integer value. The valid integer value consists of the following parts:

- (optional) plus or minus sign
- (optional) prefix (**0**) indicating octal base (applies only when the base is `8` or `0`)
- (optional) prefix (**0x** or **0X**) indicating hexadecimal base (applies only when the base is `16` or `0`)
- a sequence of digits

The set of valid values for base is {0,2,3,...,36}. The set of valid digits for base-2 integers is {0,1}, for base-3 integers is {0,1,2}, and so on. For bases larger than 10, valid digits include alphabetic characters, starting from Aa for base-11 integer, to Zz for base-36 integer. The case of the characters is ignored.

Additional numeric formats may be accepted by the currently installed C locale.

If the value of `base` is `0`, the numeric base is auto-detected: if the prefix is **0**, the base is octal, if the prefix is **0x** or **0X**, the base is hexadecimal, otherwise the base is decimal.

If the minus sign was part of the input sequence, the numeric value calculated from the sequence of digits is negated as if by unary minus in the result type.

The functions sets the pointer pointed to by `str_end` to point to the character past the last character interpreted. If `str_end` is a null pointer, it is ignored.

If the `str` is empty or does not have the expected form, no conversion is performed, and (if `str_end` is not a null pointer) the value of `str` is stored in the object pointed to by `str_end`.

Parameters

str	–	pointer to the null-terminated byte string to be interpreted
str_end	–	pointer to a pointer to character.
base	–	*base* of the interpreted integer value

Return value

- If successful, an integer value corresponding to the contents of `str` is returned.
- If the converted value falls out of range of corresponding return type, a range error occurs (setting `errno` to `ERANGE`) and `LONG_MAX`, `LONG_MIN`, `LLONG_MAX` or `LLONG_MIN` is returned.
- If no conversion can be performed, `0` is returned.

Example

Run this code

```
#include <errno.h>
#include <limits.h>
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    // parsing with error handling
    const char *p = "10 200000000000000000000000000000 30 -40 junk";
    printf("Parsing '%s':\n", p);

    for (;;)
    {
        // errno can be set to any non-zero value by a library function call
```

```
        // regardless of whether there was an error, so it needs to be cleared
        // in order to check the error set by strtol
        errno = 0;
        char *end;
        const long i = strtol(p, &end, 10);
        if (p == end)
            break;

        const bool range_error = errno == ERANGE;
        printf("Extracted '%.*s', strtol returned %ld.", (int)(end-p), p, i);
        p = end;

        if (range_error)
            printf(" Range error occurred.");

        putchar('\n');
    }

    printf("Unextracted leftover: '%s'\n\n", p);

    // parsing without error handling
    printf("\"1010\" in binary  --> %ld\n", strtol("1010", NULL, 2));
    printf("\"12\"   in octal   --> %ld\n", strtol("12",   NULL, 8));
    printf("\"A\"    in hex     --> %ld\n", strtol("A",    NULL, 16));
    printf("\"junk\" in base-36 --> %ld\n", strtol("junk", NULL, 36));
    printf("\"012\"  in auto-detected base --> %ld\n", strtol("012",  NULL, 0));
    printf("\"0xA\"  in auto-detected base --> %ld\n", strtol("0xA",  NULL, 0));
    printf("\"junk\" in auto-detected base --> %ld\n", strtol("junk", NULL, 0));
}
```

Possible output:

```
Parsing '10 2000000000000000000000000000000 30 -40 junk':
Extracted '10', strtol returned 10.
Extracted ' 2000000000000000000000000000000', strtol returned 9223372036854775807. Range erro
Extracted ' 30', strtol returned 30.
Extracted ' -40', strtol returned -40.
Unextracted leftover: ' junk'

"1010" in binary  --> 10
"12"   in octal   --> 10
"A"    in hex     --> 10
"junk" in base-36 --> 926192
"012"  in auto-detected base --> 10
"0xA"  in auto-detected base --> 10
"junk" in auto-detected base --> 0
```

References

- C11 standard (ISO/IEC 9899:2011):
  - 7.22.1.4 The strtol, strtoll, strtoul, and strtoull functions (p: 344-345)
- C99 standard (ISO/IEC 9899:1999):
  - 7.20.1.4 The strtol, strtoll, strtoul, and strtoull functions (p: 310-311)
- C89/C90 standard (ISO/IEC 9899:1990):
  - 4.10.1.5 The strtol function

See also

atoi **atol** **atoll** (C99)	converts a byte string to an integer value (function)
strtoul **strtoull** (C99)	converts a byte string to an unsigned integer value (function)
wcstol (C95) **wcstoll** (C99)	converts a wide string to an integer value (function)
	converts a wide string to an unsigned integer value

wcstoul (C95) (function)
wcstoull (C99)

C++ documentation for **strtol, strtoll**