std::**strtoul**, std::**strtoull**

Defined in header <cstdlib>		
unsigned long	strtoul(const char *str, char **str_end, int base);	
unsigned long long	strtoull(const char *str, char **str_end, int base);	(since C++11)

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

Discards any whitespace characters (as identified by calling std::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) unsigned integer number representation and converts them to an integer value. The valid unsigned 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, which applies unsigned integer wraparound rules.

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.

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

Integer value corresponding to the contents of str on success. If the converted value falls out of range of corresponding return type, range error occurs and ULONG_MAX or ULLONG_MAX is returned. If no conversion can be performed, `0` is returned.

Example

```
Run this code
```

```cpp
#include <iostream>
#include <string>
#include <errno.h>
#include <cstdlib>

int main()
{
    const char* p = "10 200000000000000000000000000000 30 -40";
    char *end;
    std::cout << "Parsing '" << p << "':\n";
    for (unsigned long i = std::strtoul(p, &end, 10);
         p != end;
         i = std::strtoul(p, &end, 10))
    {
        std::cout << "'" << std::string(p, end-p) << "' -> ";
        p = end;
        if (errno == ERANGE){
            std::cout << "range error, got ";
            errno = 0;
```

```
        }
        std::cout << i << '\n';
    }
}
```

Possible output:

```
Parsing '10 20000000000000000000000000000000 30 -40':
'10' -> 10
' 20000000000000000000000000000000' -> range error, got 18446744073709551615
' 30' -> 30
' -40' -> 18446744073709551576
```

See also

stoul (C++11) **stoull** (C++11)	converts a string to an unsigned integer (function)	
strtol **strtoll** (C++11)	converts a byte string to an integer value (function)	
strtoimax (C++11) **strtoumax** (C++11)	converts a byte string to std::intmax_t or std::uintmax_t (function)	
wcstoul **wcstoull**	converts a wide string to an unsigned integer value (function)	
strtof **strtod** **strtold**	converts a byte string to a floating point value (function)	
from_chars (C++17)	converts a character sequence to an integer or floating-point value (function)	
atoi **atol** **atoll** (C++11)	converts a byte string to an integer value (function)	
C documentation for **strtoul, strtoull**		