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

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'strtoul.3p' command

$ man strtoul.3p

STRTOUL(3P) POSIX Programmer's Manual STRTOUL(3P)

PROLOG

This manual page is part of the POSIX Programmer's Manual. The Linux implementation of this interface may differ (consult the corresponding Linux manual page for details of Linux behavior), or the interface may not be implemented on Linux.

NAME

strtoul, strtoull ? convert a string to an unsigned long

SYNOPSIS

```
#include <stdlib.h>
unsigned long strtoul(const char *restrict str,
    char **restrict endptr, int base);
unsigned long long strtoull(const char *restrict str,
    char **restrict endptr, int base);
```

DESCRIPTION

The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1?2017 defers to the ISO C standard.

These functions shall convert the initial portion of the string pointed to by str to a type unsigned long and unsigned long long representa? tion, respectively. First, they decompose the input string into three parts:

1. An initial, possibly empty, sequence of white-space characters (as specified by isspace())
2. A subject sequence interpreted as an integer represented in some radix determined by the value of base
3. A final string of one or more unrecognized characters, including the terminating NUL character of the input string

Then they shall attempt to convert the subject sequence to an unsigned integer, and return the result.

If the value of base is 0, the expected form of the subject sequence is that of a decimal constant, octal constant, or hexadecimal constant, any of which may be preceded by a '+' or '-' sign. A decimal constant begins with a non-zero digit, and consists of a sequence of decimal digits. An octal constant consists of the prefix '0' optionally fol? lowed by a sequence of the digits '0' to '7' only. A hexadecimal con? stant consists of the prefix 0x or 0X followed by a sequence of the decimal digits and letters 'a' (or 'A') to 'f' (or 'F') with values 10 to 15 respectively.

If the value of base is between 2 and 36, the expected form of the sub? ject sequence is a sequence of letters and digits representing an inte? ger with the radix specified by base, optionally preceded by a '+' or '-' sign. The letters from 'a' (or 'A') to 'z' (or 'Z') inclusive are ascribed the values 10 to 35; only letters whose ascribed values are less than that of base are permitted. If the value of base is 16, the characters 0x or 0X may optionally precede the sequence of letters and digits, following the sign if present.

The subject sequence is defined as the longest initial subsequence of the input string, starting with the first non-white-space character that is of the expected form. The subject sequence shall contain no characters if the input string is empty or consists entirely of white-space characters, or if the first non-white-space character is other than a sign or a permissible letter or digit.

If the subject sequence has the expected form and the value of base is 0, the sequence of characters starting with the first digit shall be

interpreted as an integer constant. If the subject sequence has the ex?
pected form and the value of base is between 2 and 36, it shall be used
as the base for conversion, ascribing to each letter its value as given
above. If the subject sequence begins with a <hyphen-minus>, the value
resulting from the conversion shall be negated. A pointer to the final
string shall be stored in the object pointed to by endptr, provided
that endptr is not a null pointer.

In other than the C or POSIX locale, additional locale-specific subject
sequence forms may be accepted.

If the subject sequence is empty or does not have the expected form, no
conversion shall be performed; the value of str shall be stored in the
object pointed to by endptr, provided that endptr is not a null
pointer.

These functions shall not change the setting of errno if successful.

Since 0, {ULONG_MAX}, and {ULLONG_MAX} are returned on error and are
also valid returns on success, an application wishing to check for er?
ror situations should set errno to 0, then call strtoul() or str?
toull(), then check errno.

RETURN VALUE

Upon successful completion, these functions shall return the converted
value, if any. If no conversion could be performed, 0 shall be returned
and errno may be set to [EINVAL].

If the value of base is not supported, 0 shall be returned and errno
shall be set to [EINVAL].

If the correct value is outside the range of representable values,
{ULONG_MAX} or {ULLONG_MAX} shall be returned and errno set to
[ERANGE].

ERRORS

These functions shall fail if:

EINVAL The value of base is not supported.

ERANGE The value to be returned is not representable.

These functions may fail if:

EINVAL No conversion could be performed.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

Since the value of *endptr is unspecified if the value of base is not supported, applications should either ensure that base has a supported value (0 or between 2 and 36) before the call, or check for an [EINVAL] error before examining *endptr.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

fscanf(), isalpha(), strtod(), strtol()

The Base Definitions volume of POSIX.1?2017, <stdlib.h>

COPYRIGHT