Split the Elements of a Character Vector

Description

Split the elements of a character vector `x` into substrings according to the matches to substring `split` within them.

Usage

```
strsplit(x, split, fixed = FALSE, perl = FALSE, useBytes = FALSE)
```

Arguments

`x`	character vector, each element of which is to be split. Other inputs, including a factor, will give an error.
`split`	character vector (or object which can be coerced to such) containing regular expression(s) (unless `fixed = TRUE`) to use for splitting. If empty matches occur, in particular if `split` has length 0, `x` is split into single characters. If `split` has length greater than 1, it is re-cycled along `x`.
`fixed`	logical. If `TRUE` match `split` exactly, otherwise use regular expressions. Has priority over `perl`.
`perl`	logical. Should Perl-compatible regexps be used?
`useBytes`	logical. If `TRUE` the matching is done byte-by-byte rather than character-by-character, and inputs with marked encodings are not converted. This is forced (with a warning) if any input is found which is marked as `"bytes"` (see `Encoding`).

Details

Argument `split` will be coerced to character, so you will see uses with `split = NULL` to mean `split = character(0)`, including in the examples below.

Note that splitting into single characters can be done *via* `split = character(0)` or `split = ""`; the two are equivalent. The definition of 'character' here depends on the locale: in a single-byte locale it is a byte, and in a multi-byte locale it is the unit represented by a 'wide character' (almost always a Unicode code point).

A missing value of `split` does not split the corresponding element(s) of `x` at all.

The algorithm applied to each input string is

```
    repeat {
        if the string is empty
            break.
        if there is a match
            add the string to the left of the match to the output.
            remove the match and all to the left of it.
        else
            add the string to the output.
            break.
    }
```

Note that this means that if there is a match at the beginning of a (non-empty) string, the first element of the output is `""`, but if there is a match at the end of the string, the output is the same as with the match removed.

Invalid inputs in the current locale are warned about up to 5 times.

Value

A list of the same length as `x`, the `i`-th element of which contains the vector of splits of `x[i]`.

If any element of `x` or `split` is declared to be in UTF-8 (see `Encoding`), all non-ASCII character strings in the result will be in UTF-8 and have their encoding declared as UTF-8. For `perl = TRUE, useBytes = FALSE` all non-ASCII strings in a multibyte locale are translated to UTF-8.

See Also

`paste` for the reverse, `grep` and `sub` for string search and manipulation; also `nchar`, `substr`.

'regular expression' for the details of the pattern specification.

Option `PCRE_use_JIT` controls the details when `perl = TRUE`.

Examples

```
noquote(strsplit("A text I want to display with spaces", NULL)[[1]])

x <- c(as = "asfef", qu = "qwerty", "yuiop[", "b", "stuff.blah.yech")
# split x on the letter e
strsplit(x, "e")

unlist(strsplit("a.b.c", "."))
## [1] "" "" "" "" ""
## Note that 'split' is a regexp!
## If you really want to split on '.', use
unlist(strsplit("a.b.c", "[.]"))
## [1] "a" "b" "c"
```

```
## or
unlist(strsplit("a.b.c", ".", fixed = TRUE))

## a useful function: rev() for strings
strReverse <- function(x)
        sapply(lapply(strsplit(x, NULL), rev), paste, collapse = "")
strReverse(c("abc", "Statistics"))

## get the first names of the members of R-core
a <- readLines(file.path(R.home("doc"),"AUTHORS"))[-(1:8)]
a <- a[(0:2)-length(a)]
(a <- sub(" .*","", a))
# and reverse them
strReverse(a)

## Note that final empty strings are not produced:
strsplit(paste(c("", "a", ""), collapse="#"), split="#")[[1]]
# [1] ""  "a"
## and also an empty string is only produced before a definite match:
strsplit("", " ")[[1]]    # character(0)
strsplit(" ", " ")[[1]]   # [1] ""
```
