SEP #26 : strsplit(), add the capability to split a string on a pattern.

Title: strsplit(), add the capability to split a string on a pattern.
Version: 1.0
Author: Pierre MARECHAL (pierre.marechal@scilab.org)
Review:
Commented:
State: Proposal
Scilab-Version: 5.2
Vote:
Created: 6th May 2009

Abstract

The strsplit() function is used to split a string into smaller sections.

Actually, strsplit(**str**,**ind**) splits the string "**str**" into a vector of strings at the points given by the indices in "**ind**" and returns a column vector of string.

This SEP proposes to add the capability to split a string on a single character, a group of characters or a regular expression (a pattern).

The present document also suggests to add a third input argument "**limit**". If "**limit**" is set, the returned array will contain a maximum of "**limit**" elements with the last element containing the whole rest of string.

Rational

With the current behavior, Scilab user has to know where he will split the string and in how many sub-sections before calling strsplit() function. This can be very limiting when parsing strings.

For example, to parse the following "crappy" but allowed ini file. Spaces have been replaced by dots and tabulations by arrows.

```
server.name.....=.scilab.org
port..→→=.143
database.=→members
```

With the current behavior, for each line,

```
// Detect the “=” index
equal_ind = regexp(line,”/=/”,”o”)

// Split the line
splitted_line = split(line,equal_ind)

// Remove the “=” character
splitted_line = strsubst(splitted_line,”=”,””)

// Remove the leading and trailing white space characters
splitted_line = stripblanks(splitted_line,%T);
```

With the proposed improvement, this can be packed in one small line:

```
splitted_line = strsplit(line,”/\s*=\s*/”);
```

Return value

```
--> strsplit(str,pattern,limit)
```

Returns an column vector of strings, each of which is a substring of str formed by splitting it on boundaries formed by the case-sensitive regular expression pattern . If there are n occurrences of pattern , the returned array will contain *n+1* items. For example, if there is no occurrence of pattern , an array with only one element will be returned. Of course, this is also true if str is empty.

If limit is set, the returned array will contain a maximum of limit elements with the last element containing the whole rest of string.

pattern string is treated as regular expression if it starts with “/” **and** ends with “/” or “/i”.

Example Usage

Example 1. Splitting on a character

A common use of split() is when parsing data from a file or from another program. In this example, we will split the string on the comma ','. Note that you typically should not use split() to parse CSV (comma separated value) files in case there are commas in your data: use Text::CSV instead.

```
-->strsplit(“abcdef,ghijkl,mnopqr,stuvw,xyz”,”,”)
 ans  =
! abcdef  !
! ghijkl  !
! mnopqr  !
! stuvw   !
! xyz     !
```

Example 2. Splitting on a string

In the same way you use a character to split, you can use a string. In this example, the data is separated by three tildas '~~~'.

```
-->strsplit("abcdef~~~ghijkl~~~mnopqr~~~stuvw~~~xyz","~~~")
 ans  =
! abcdef  !
! ghijkl  !
! mnopqr  !
! stuvw   !
! xyz     !
```

Example 3. Splitting on a pattern

In some cases, you may want to split the string on a pattern (regular expression) or a type of character. We'll assume here that you know a little about regular expressions.

In this example we will split on any integer:

```
-->strsplit("abcdef2ghijkl3mnopqr6stuvw7xyz","/\d+/")
 ans  =
! abcdef  !
! ghijkl  !
! mnopqr  !
! stuvw   !
! xyz     !
```

In this example we will split on any tabulation:

```
-->strsplit("abcdef   ghijkl  mnopqr  stuvw   xyz","/\t/")
 ans  =
! abcdef  !
! ghijkl  !
! mnopqr  !
! stuvw   !
! xyz     !
```

Change log

`1.0` — Pierre Marechal <pierre.marechal@scilab.org> - `Initial Revision`

Copyright