

Description

`pathjoin(path1, path2)` forms, logically speaking, *path1/path2*, but does so in the appropriate style. For instance, *path1* might be a URL and *path2* a Windows *dirname\filename*, and the two paths will, even so, be joined correctly. All issues of whether *path1* ends with a directory separator, *path2* begins with one, etc., are handled automatically.

`pathsplit(path, path1, path2)` performs the inverse operation, removing the last element of the path (which is typically a filename) and storing it in *path2* and storing the rest in *path1*.

`pathbasename(path)` returns the last element of *path*.

`pathsuffix(path)` returns the file suffix, with leading dot, if there is one, and returns "" otherwise. For instance, `pathsuffix("this\that.ado")` returns ".ado".

`pathrmsuffix(path)` returns *path* with the suffix removed, if there was one. For instance, `pathrmsuffix("this\that.ado")` returns "this\that".

`pathisurl(path)` returns 1 if *path* is a URL and 0 otherwise.

`pathisabs(path)` returns 1 if *path* is absolute and 0 if relative. `c:\this` is an absolute path. `this\that` is a relative path. URLs are considered to be absolute.

`pathasciisuffix(path)` and `pathstatastatusuffix(path)` are more for StataCorp use than anything else. `pathasciisuffix()` returns 1 if the file is known to be text, based on its file suffix. StataCorp uses this function in Stata's `net` command to decide whether end-of-line characters, which differ across operating systems, should be modified during downloads. `pathstatastatusuffix()` is the function used by Stata's `net` and `update` commands to decide whether a file belongs in the official directories. `pathstatastatusuffix("example.ado")` is true, but `pathstatastatusuffix("example.do")` is false because do-files do not go in system directories.

`pathlist(dirlist)` returns a row vector, each element of which contains an element of a semicolon-separated path list *dirlist*. For instance, `pathlist("a;b;c")` returns ("a", "b", "c").

`pathlist()` without arguments returns `pathlist(c("adopath"))`, the broken-out elements of the official Stata ado-path.

`pathsubsysdir(pathlist)` returns *pathlist* with any elements that are Stata system directories' short-hands, such as PLUS, PERSONAL, substituted with the actual directory names. For instance, the right way to obtain the official directories over which Stata searches for files is `pathsubsysdir(pathlist())`.

`pathsearchlist(fn)` returns a row vector. The elements are full paths/filenames specifying all the locations, in order, where Stata would look for *fn* along the official Stata ado-path.

`pathresolve(basepath, path)` returns a resolved path of *path* against *basepath*. For example, `pathresolve("c:/test", "../test1")` returns "c:/test1".

`pathgetparent(path)` returns the parent path of *path*. For example, `pathgetparent("c:/test/test.do")` returns "c:/test". Neither `c:/test` nor `c:/test/test.do` need exist.

Syntax

<i>string scalar</i>	<code>pathjoin(<i>string scalar path1</i> , <i>string scalar path2</i>)</code>
<i>void</i>	<code>pathsplitt(<i>string scalar path</i> , <i>path1</i> , <i>path2</i>)</code>
<i>string scalar</i>	<code>pathbasename(<i>string scalar path</i>)</code>
<i>string scalar</i>	<code>pathsuffix(<i>string scalar path</i>)</code>
<i>string scalar</i>	<code>pathrmsuffix(<i>string scalar path</i>)</code>
<i>real scalar</i>	<code>pathisurl(<i>string scalar path</i>)</code>
<i>real scalar</i>	<code>pathisabs(<i>string scalar path</i>)</code>
<i>real scalar</i>	<code>pathasciisuffix(<i>string scalar path</i>)</code>
<i>real scalar</i>	<code>pathstata suffix(<i>string scalar path</i>)</code>
<i>string rowvector</i>	<code>pathlist(<i>string scalar dirlist</i>)</code>
<i>string rowvector</i>	<code>pathlist()</code>
<i>string rowvector</i>	<code>pathsubsysdir(<i>string rowvector pathlist</i>)</code>
<i>string rowvector</i>	<code>pathsearchlist(<i>string scalar fn</i>)</code>
<i>string scalar</i>	<code>pathresolve(<i>string scalar basepath</i> , <i>path</i>)</code>
<i>string scalar</i>	<code>pathgetparent(<i>string scalar path</i>)</code>

Remarks and examples

Using these functions, you are more likely to produce code that works correctly regardless of operating system.

`pathremove()` returns an error code if you do not have the necessary permission to remove the directory or its contents.

`pathresolve(basepath , path)` simply returns *path* if it is an absolute path.

`pathresolve(basepath , path)`'s behavior is undefined if *basepath* has no root element, for example, `../`.

Conformability

`pathjoin(path1, path2):`

path1: 1 × 1
path2: 1 × 1
result: 1 × 1

`pathsplit(path, path1, path2):`

input:

path: 1 × 1

output:

path1: 1 × 1
path2: 1 × 1

`pathbasename(path), pathsuffix(path), pathrmsuffix(path):`

path: 1 × 1
result: 1 × 1

`pathisurl(path), pathisabs(path), pathasciisuffix(path), pathstatusuffix(path):`

path: 1 × 1
result: 1 × 1

`pathlist(dirlist):`

dirlist: 1 × 1 (optional)
result: 1 × *k*

`pathsubsysdir(pathlist):`

pathlist: 1 × *k*
result: 1 × *k*

`pathsearchlist(fn):`

fn: 1 × 1
result: 1 × *k*

`pathresolve(basepath, path):`

basepath: 1 × 1
path: 1 × 1
result: 1 × 1

`pathgetparent(path):`

path: 1 × 1
result: 1 × 1

Diagnostics

All routines abort with error if the path is too long for the operating system.

Also see

[M-4] **IO** — I/O functions

Stata, Stata Press, Mata, NetCourse, and NetCourseNow are registered trademarks of StataCorp LLC. Stata and Stata Press are registered trademarks with the World Intellectual Property Organization of the United Nations. StataNow is a trademark of StataCorp LLC. Other brand and product names are registered trademarks or trademarks of their respective companies. Copyright © 1985–2025 StataCorp LLC, College Station, TX, USA. All rights reserved.



For suggested citations, see the FAQ on [citing Stata documentation](#).