

## pathjoin() — File path manipulation

Description Diagnostics	Syntax Also see	Remarks and examples	Conformability
----------------------------	--------------------	----------------------	----------------

## 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.

## Syntax

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

## Remarks and examples

[stata.com](https://www.stata.com)

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

## Conformability

`pathjoin(path1, path2):`

*path1:*  $1 \times 1$

*path2:*  $1 \times 1$

*result:*  $1 \times 1$

`pathsplit(path, path1, path2):`

*input:*

*path:*  $1 \times 1$

*output:*

*path1:*  $1 \times 1$

*path2:*  $1 \times 1$

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

*path:*  $1 \times 1$

*result:*  $1 \times 1$

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

*path:*  $1 \times 1$

*result:*  $1 \times 1$

`pathlist(dirlist):`

*dirlist:*  $1 \times 1$  (optional)

*result:*  $1 \times k$

`pathsysdir(pathlist):`

*pathlist:*  $1 \times k$

*result:*  $1 \times k$

`pathsearchlist(fn):`

*fn:*  $1 \times 1$

*result:*  $1 \times k$

## Diagnostics

All routines abort with error if the path is too long for the operating system; nothing else causes abort with error.

## Also see

[M-4] [io](#) — I/O functions