Title

stata.com

IO — I/O functions

Contents Description Remarks and examples Reference Also see

Contents

File I/O

fopen( )	`fopen()`	open file
	`fclose()`	close file
	`fget()`	read line of text file
	`fgetnl()`	same, but include newline character
	`fread()`	read *k* bytes of binary file
	`fput()`	write line into text file
	`fwrite()`	write *k* bytes into binary file
	`fgetmatrix()`	read matrix
	`fputmatrix()`	write matrix
	`fstatus()`	status of last I/O command
	`ftell()`	report location in file
	`fseek()`	seek to location in file
	`ftruncate()`	truncate file at current position
ferrortext( )	`ferrortext()`	error text of file error code
	`freturncode()`	return code of file error code
bufio( )	`bufio()`	initialize buffer
	`bufbyteorder()`	reset (specify) byte order
	`bufmissingvalue()`	reset (specify) missing-value encoding
	`bufput()`	copy into buffer
	`bufget()`	copy from buffer
	`fbufput()`	copy into and write buffer
	`fbufget()`	read and copy from buffer
	`bufbfmtlen()`	utility routine
	`bufbfmtisnum()`	utility routine
xl( )	`xl()`	Excel file I/O class
_docx*( )	`_docx*()`	generate Office Open XML file
Pdf*( )	`Pdf*()`	create a PDF file

Filename & path manipulation

pathjoin( )	`pathjoin()`	join paths
	`pathsplit()`	split paths
	`pathbasename()`	path basename
	`pathsuffix()`	file suffix
	`pathrmsuffix()`	remove file suffix
	`pathisurl()`	whether path is URL
	`pathisabs()`	whether path is absolute
	`pathasciisuffix()`	whether file is text
	`pathstatasuffix()`	whether file is Stata
	`pathlist()`	process path list
	`pathsubsysdir()`	substitute for system directories
	`pathsearchlist()`	path list to search for file
	`pathresolve()`	resolve a relative path
	`pathgetparent()`	get the parent path

Description

The above functions have to do with

1. Displaying output at the terminal.
2. Reading and writing data in a file.

Remarks and examples

stata.com

To display the contents of a scalar, vector, or matrix, it is sufficient merely to code the identity of the scalar, vector, or matrix:

```
: x
                 1             2             3             4
  1  .1369840784    .643220668    .5578016951    .6047949435
```

You can follow this approach even in programs:

```
function example()
{
        ...
        "i am about to calculate the result"
        ...
        "the result is"
        b
}
```

On the other hand, `display()` and `printf()` (see [M-5] **display( )** and [M-5] **printf( )**) will allow you to exercise more control over how the output looks.

Changing the subject: you will find that many I/O functions come in two varieties: with and without an underscore in front of the name, such as `_fopen()` and `fopen()`. As always, functions that begin with an underscore are generally silent about their work and return flags indicating their success or failure. Functions that omit the underscore abort and issue the appropriate error message when things go wrong.

Reference

Gould, W. W. 2009. Mata Matters: File processing. *Stata Journal* 9: 599–620.

Also see

[M-4] **Intro** — Categorical guide to Mata functions