

## io — I/O functions

[Contents](#)   [Description](#)   [Remarks and examples](#)   [Reference](#)   [Also see](#)

## Contents

[M-5] Manual entry	Function	Purpose
<b>Console output</b>		
<b>printf()</b>	printf() sprintf()	display display into string
<b>errprintf()</b>	errprintf()	display error message
<b>display()</b>	display()	display text interpreting SMCL
<b>displayas()</b>	displayas()	set whether output is displayed
<b>displayflush()</b>	displayflush()	flush terminal output buffer
<b>liststruct()</b>	liststruct()	list structure's contents
<b>more()</b>	more() setmore() setmoreonexit()	create <code>—more—</code> condition query or set more on or off set more on or off on exit

**File directories**

<b>direxists()</b>	direxists()	whether directory exists
<b>dir()</b>	dir()	file list
<b>chdir()</b>	pwd() chdir() mkdir() rmdir()	obtain current working directory change current working directory make new directory remove directory

**File management**

<b>findfile()</b>	findfile()	find file
<b>fileexists()</b>	fileexists()	whether file exists
<b>cat()</b>	cat()	read file into string matrix
<b>unlink()</b>	unlink()	erase file
<b>adosubdir()</b>	adosubdir()	obtain ado-subdirectory for file

File I/O
----------

---

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

Filename & path manipulation
------------------------------

---

<b>pathjoin()</b>	<code>pathjoin()</code> <code>pathsplrit()</code> <code>pathbasename()</code> <code>pathsuffix()</code> <code>pathrmsuffix()</code> <code>pathisurl()</code> <code>pathisabs()</code> <code>pathasciisuffix()</code> <code>pathstata suffix()</code> <code>pathlist()</code> <code>pathsbsysdir()</code> <code>pathsearchlist()</code>	join paths split paths path basename file suffix remove file suffix whether path is URL whether path is absolute whether file is text whether file is Stata process path list substitute for system directories path list to search for file
-------------------	---	---

---

## 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](http://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