ferrortext() — Text and return code of file error code

Description

`ferrortext(`*ec*`)` returns the text associated with a file error code returned by, for instance, `_fopen()`, `_fwrite()`, `fstatus()`, or any other file-processing functions that return an error code. See [M-5] **fopen()**.

`freturncode(`*ec*`)` returns the Stata return code associated with the file error code.

Syntax

string scalar `ferrortext(`*real scalar ec*`)`

real scalar `freturncode(`*real scalar ec*`)`

Remarks and examples

Most file-processing functions abort with error if something goes wrong. You attempt to read a nonexisting file, or attempt to write a read-only file, and the file functions you use to do that, usually documented in [M-5] **fopen()**, abort with error. Abort with error means not only that the file function you called stops when the error arises but also that the calling function is aborted. The user is presented with a traceback log documenting what went wrong.

Sometimes you will want to write code to handle such errors for itself. For instance, you are writing a subroutine for a command to be used in Stata and, if the file does not exist, you want the subroutine to present an informative error message and exit without a traceback log but with a nonzero return code. Or in another application, if the file does not exist, that is not an error at all; your code will create the file.

Most file-processing functions have a corresponding underscore function that, rather than aborting, returns an error code when things go wrong. `fopen()` opens a file or aborts with error. `_fopen()` opens a file or returns an error code. The error code is sufficient for your program to take the appropriate action. One uses the underscore functions when the calling program will deal with any errors that might arise.

Let's take the example of simply avoiding traceback logs. If you code

```
fh = fopen(filename, "r")
```

and the file does not exist, execution aborts and you see a traceback log. If you code

```
if ((fh = _fopen(filename, "r"))<0) {
        errprintf("%s\n", ferrortext(fh))
        exit(freturncode(fh))
}
```

execution still stops if the file does not exist, but this time, it stops because you coded `exit()`. You still see an error message, but this time, you see the message because you coded `errprintf()`. No traceback log is produced because you did not insert code to produce one. You could have coded `_exit()` if you wanted one.

The file error codes and the messages associated with them are

Negative code	Meaning
0	all is well
−1	end of file *(this code is usually not an error)*
−601	file not found
−602	file already exists
−603	file could not be opened
−608	file is read-only
−610	file not Stata format
−612	unexpected end of file
−630	web files not supported in this version of Stata
−631	host not found
−632	web file not allowed in this context
−633	may not write to web file
−639	file transmission error—checksums do not match
−660	proxy host not found
−662	proxy server refused request to send
−663	remote connection to proxy failed
−665	could not set socket nonblocking
−667	wrong version of `winsock.dll`
−668	could not find valid `winsock.dll` or `astsys0.lib`
−669	invalid URL
−670	invalid network port number
−671	unknown network protocol
−672	server refused to send file
−673	authorization required by server
−674	unexpected response from server
−675	server reported error
−676	server refused request to send
−677	remote connection failed—see `r(677)` for troubleshooting
−678	could not open local network socket
−679	unexpected web error

−680	could not find valid `odbc32.dll`
−681	too many open files
−682	could not connect to ODBC data source name
−683	could not fetch variable in ODBC table
−684	could not find valid `dlxabi32.dll`
−691	I/O error
−699	insufficient disk space
−3601	invalid file handle
−3602	invalid filename
−3611	too many open files
−3621	attempt to write read-only file
−3622	attempt to read write-only file
−3623	attempt to seek append-only file
−3698	file seek error

File error codes are usually negative, but neither `ferrortext(`*ec*`)` nor `freturncode(`*ec*`)` cares whether *ec* is of the positive or negative variety.

Conformability

`ferrortext(`*ec*`)`, `freturncode(`*ec*`)`:

ec: 1×1

result: 1×1

Diagnostics

`ferrortext(`*ec*`)` and `freturncode(`*ec*`)` treat $ec = -1$ (end of file) the same as $ec = 612$ (unexpected end of file). Code −1 usually is not an error; it just denotes end of file. It is assumed that you will not call `ferrortext()` and `freturncode()` in such cases. If you do call the functions here, it is assumed that you mean that the end of file was unexpected.

Also see

[M-5] **fopen( )** — File I/O

[M-4] **io** — I/O functions