File Objects

These APIs are a minimal emulation of the Python 2 C API for built-in file objects, which used to rely on the buffered I/O (`FILE*`) support from the C standard library. In Python 3, files and streams use the new `io` module, which defines several layers over the low-level unbuffered I/O of the operating system. The functions described below are convenience C wrappers over these new APIs, and meant mostly for internal error reporting in the interpreter; third-party code is advised to access the `io` APIs instead.

PyFile_FromFd(int *fd*, const char **name*, const char **mode*, int *buffering*, const char **encoding*, const char **errors*, const char **newline*, int *closefd*)

Create a Python file object from the file descriptor of an already opened file *fd*. The arguments *name*, *encoding*, *errors* and *newline* can be *NULL* to use the defaults; *buffering* can be *-1* to use the default. *name* is ignored and kept for backward compatibility. Return *NULL* on failure. For a more comprehensive description of the arguments, please refer to the `io.open()` function documentation.

> **Warning:** Since Python streams have their own buffering layer, mixing them with OS-level file descriptors can produce various issues (such as unexpected ordering of data).

Changed in version 3.2: Ignore *name* attribute.

int **PyObject_AsFileDescriptor**(PyObject **p*)

Return the file descriptor associated with *p* as an `int`. If the object is an integer, its value is returned. If not, the object's `fileno()` method is called if it exists; the method must return an integer, which is returned as the file descriptor value. Sets an exception and returns `-1` on failure.

PyObject* **PyFile_GetLine**(PyObject **p*, int *n*)

Return value: New reference.

Equivalent to `p.readline([n])`, this function reads one line from the object *p*. *p* may be a file object or any object with a `readline()` method. If *n* is `0`, exactly one line is read, regardless of the length of the line. If *n* is greater than `0`, no more than *n* bytes will be read from the file; a partial line can be returned. In both cases, an empty string is returned if the end of the file is reached immediately. If *n* is less than `0`, however, one line is read regardless of length, but `EOFError` is raised if the end of the file is reached immediately.

int **PyFile_WriteObject**(PyObject **obj*, PyObject **p*, int *flags*)

Write object *obj* to file object *p*. The only supported flag for *flags* is `Py_PRINT_RAW`; if given, the `str()` of the object is written instead of the `repr()`. Return `0` on success or `-1` on failure; the appropriate exception will be set.

int **PyFile_WriteString**(const char **s*, PyObject **p*)

Write string *s* to file object *p*. Return `0` on success or `-1` on failure; the appropriate exception will be set.