Previous: 6.1.2 File Object Creation **Up:** 6.1 os **Next:** 6.1.4 Files and Directories

6.1.3 File Descriptor Operations

These functions operate on I/O streams referred to using file descriptors.

`close`(*fd*)
Close file descriptor *fd*. Availability: Macintosh, UNIX, Windows.

Note: this function is intended for low-level I/O and must be applied to a file descriptor as retu
`()` or `pipe()`. To close a ``file object'' returned by the built-in function `open()` or by `popen()`
use its `close()` method.

`dup`(*fd*)
Return a duplicate of file descriptor *fd*. Availability: Macintosh, UNIX, Windows.

`dup2`(*fd*, *fd2*)
Duplicate file descriptor *fd* to *fd2*, closing the latter first if necessary. Availability: UNIX, Win

`fdatasync`(*fd*)
Force write of file with filedescriptor *fd* to disk. Does not force update of metadata. Availabili

`fpathconf`(*fd*, *name*)
Return system configuration information relevant to an open file. *name* specifies the configura
retrieve; it may be a string which is the name of a defined system value; these names are speci
number of standards (POSIX.1, UNIX 95, UNIX 98, and others). Some platforms define additi
well. The names known to the host operating system are given in the `pathconf_names` dictior
configuration variables not included in that mapping, passing an integer for *name* is also accep
Availability: UNIX.

If *name* is a string and is not known, `ValueError` is raised. If a specific value for *name* is not

the host system, even if it is included in `pathconf_names`, an `OSError` is raised with `errno.E` error number.

`fstat`(*fd*)
Return status for file descriptor *fd*, like `stat()`. Availability: UNIX, Windows.

`fstatvfs`(*fd*)
Return information about the filesystem containing the file associated with file descriptor *fd*, l Availability: UNIX.

`fsync`(*fd*)
Force write of file with filedescriptor *fd* to disk. On UNIX, this calls the native `fsync()` functi Windows, the MS `_commit()` function.

If you're starting with a Python file object *f*, first do `f.flush()`, and then do `os.fsync(f.fil` ensure that all internal buffers associated with *f* are written to disk. Availability: UNIX, and W in 2.2.3.

`ftruncate`(*fd, length*)
Truncate the file corresponding to file descriptor *fd*, so that it is at most *length* bytes in size. A UNIX.

`isatty`(*fd*)
Return `True` if the file descriptor *fd* is open and connected to a tty(-like) device, else `False`. A UNIX.

`lseek`(*fd, pos, how*)
Set the current position of file descriptor *fd* to position *pos*, modified by *how*: `0` to set the posi the beginning of the file; `1` to set it relative to the current position; `2` to set it relative to the enc Availability: Macintosh, UNIX, Windows.

`open`(*file, flags*[, *mode*])

Open the file *file* and set various flags according to *flags* and possibly its mode according to *m*
default *mode* is `0777` (octal), and the current umask value is first masked out. Return the file d
newly opened file. Availability: Macintosh, UNIX, Windows.

For a description of the flag and mode values, see the C run-time documentation; flag constan
`O_RDONLY` and `O_WRONLY`) are defined in this module too (see below).

Note: this function is intended for low-level I/O. For normal usage, use the built-in function `o`
returns a ``file object'' with `read()` and `write()` methods (and many more).

`openpty`()

Open a new pseudo-terminal pair. Return a pair of file descriptors (*`master, slave`*) for the p
respectively. For a (slightly) more portable approach, use the `pty` module. Availability: Some
UNIX.

`pipe`()

Create a pipe. Return a pair of file descriptors (*`r, w`*) usable for reading and writing, respecti
Availability: UNIX, Windows.

`read`(*fd*, *n*)

Read at most *n* bytes from file descriptor *fd*. Return a string containing the bytes read. If the e
referred to by *fd* has been reached, an empty string is returned. Availability: Macintosh, UNIX

Note: this function is intended for low-level I/O and must be applied to a file descriptor as retu
`()` or `pipe()`. To read a ``file object'' returned by the built-in function `open()` or by `popen()`
`sys.stdin`, use its `read()` or `readline()` methods.

`tcgetpgrp`(*fd*)

Return the process group associated with the terminal given by *fd* (an open file descriptor as r
`()`). Availability: UNIX.

`tcsetpgrp`(*fd*, *pg*)

Set the process group associated with the terminal given by *fd* (an open file descriptor as retur
to *pg*. Availability: UNIX.

ttyname(*fd*)
Return a string which specifies the terminal device associated with file-descriptor *fd*. If *fd* is n
with a terminal device, an exception is raised. Availability: UNIX.

write(*fd*, *str*)
Write the string *str* to file descriptor *fd*. Return the number of bytes actually written. Availabil
UNIX, Windows.

Note: this function is intended for low-level I/O and must be applied to a file descriptor as retu
`()` or `pipe()`. To write a ``file object'' returned by the built-in function `open()` or by `popen()`
or `sys.stdout` or `sys.stderr`, use its `write()` method.

The following data items are available for use in constructing the *flags* parameter to the `open()` func

O_RDONLY

O_WRONLY

O_RDWR

O_NDELAY

O_NONBLOCK

O_APPEND

O_DSYNC

O_RSYNC

O_SYNC

O_NOCTTY

O_CREAT

O_EXCL

O_TRUNC
Options for the *flag* argument to the `open()` function. These can be bit-wise OR'd together. A
Macintosh, UNIX, Windows.

O_BINARY
Option for the *flag* argument to the `open()` function. This can be bit-wise OR'd together with
above. Availability: Macintosh, Windows.

O_NOINHERIT

O_SHORT_LIVED

O_TEMPORARY

O_RANDOM

O_SEQUENTIAL

O_TEXT
Options for the *flag* argument to the `open()` function. These can be bit-wise OR'd together. A
Windows.