

:mod:`fcntl` --- The `fcntl` and `ioctl` system calls

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) fcntl.rst, line 1); [backlink](#)

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) fcntl.rst, line 4)

Unknown directive type "module".

```
.. module:: fcntl
   :platform: Unix
   :synopsis: The fcntl() and ioctl() system calls.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) fcntl.rst, line 8)

Unknown directive type "sectionauthor".

```
.. sectionauthor:: Jaap Vermeulen
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) fcntl.rst, line 10)

Unknown directive type "index".

```
.. index::
   pair: UNIX; file control
   pair: UNIX; I/O control
```

This module performs file control and I/O control on file descriptors. It is an interface to the `:c:func:`fcntl`` and `:c:func:`ioctl`` Unix routines. For a complete description of these calls, see `:manpage:`fcntl(2)`` and `:manpage:`ioctl(2)`` Unix manual pages.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) fcntl.rst, line 16); [backlink](#)

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) fcntl.rst, line 16); [backlink](#)

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) fcntl.rst, line 16); [backlink](#)

Unknown interpreted text role "manpage".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) fcntl.rst, line 16); [backlink](#)

Unknown interpreted text role "manpage".

All functions in this module take a file descriptor `fd` as their first argument. This can be an integer file descriptor, such as returned by `sys.stdin.fileno()`, or an `:class:`io.IOBase`` object, such as `sys.stdin` itself, which provides a `:meth:`~io.IOBase.fileno`` that returns a genuine file descriptor.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) fcntl.rst, line 21); [backlink](#)

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) fcntl.rst, line 21); [backlink](#)

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) fcntl.rst, line 27)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.3
   Operations in this module used to raise an :exc:`IOError` where they now
   raise an :exc:`OSError`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) fcntl.rst, line 31)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.8
   The fcntl module now contains ``F_ADD_SEALS``, ``F_GET_SEALS``, and
   ``F_SEAL_*`` constants for sealing of :func:`os.memfd_create` file
   descriptors.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) fcntl.rst, line 36)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.9
   On macOS, the fcntl module exposes the ``F_GETPATH`` constant, which obtains
   the path of a file from a file descriptor.
   On Linux (>=3.15), the fcntl module exposes the ``F_OFD_GETLK``, ``F_OFD_SETLK``
   and ``F_OFD_SETLKW`` constants, which working with open file description locks.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) fcntl.rst, line 42)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.10
   On Linux >= 2.6.11, the fcntl module exposes the ``F_GETPIPE_SZ`` and
   ``F_SETPIPE_SZ`` constants, which allow to check and modify a pipe's size
   respectively.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) fcntl.rst, line 47)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.11
   On FreeBSD, the fcntl module exposes the ``F_DUP2FD`` and ``F_DUP2FD_CLOEXEC``
   constants, which allow to duplicate a file descriptor, the latter setting
   ``FD_CLOEXEC`` flag in addition.
```

The module defines the following functions:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) fcntl.rst, line 55)

Unknown directive type "function".

```
.. function:: fcntl(fd, cmd, arg=0)

   Perform the operation *cmd* on file descriptor *fd* (file objects providing
   a :meth:`~io.IOBase.fileno` method are accepted as well). The values used
```

for `*cmd*` are operating system dependent, and are available as constants in the `:mod:`fcntl`` module, using the same names as used in the relevant C header files. The argument `*arg*` can either be an integer value, or a `:class:`bytes`` object. With an integer value, the return value of this function is the integer return value of the C `:c:func:`fcntl`` call. When the argument is bytes it represents a binary structure, e.g. created by `:func:`struct.pack``. The binary data is copied to a buffer whose address is passed to the C `:c:func:`fcntl`` call. The return value after a successful call is the contents of the buffer, converted to a `:class:`bytes`` object. The length of the returned object will be the same as the length of the `*arg*` argument. This is limited to 1024 bytes. If the information returned in the buffer by the operating system is larger than 1024 bytes, this is most likely to result in a segmentation violation or a more subtle data corruption.

If the `:c:func:`fcntl`` fails, an `:exc:`OSError`` is raised.

```
.. audit-event:: fcntl.fcntl fd,cmd,arg fcntl.fcntl
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main) (Doc) (library) fcntl.rst, line 79)

Unknown directive type "function".

```
.. function:: ioctl(fd, request, arg=0, mutate_flag=True)
```

This function is identical to the `:func:`~fcntl.fcntl`` function, except that the argument handling is even more complicated.

The `*request*` parameter is limited to values that can fit in 32-bits. Additional constants of interest for use as the `*request*` argument can be found in the `:mod:`termios`` module, under the same names as used in the relevant C header files.

The parameter `*arg*` can be one of an integer, an object supporting the read-only buffer interface (like `:class:`bytes``) or an object supporting the read-write buffer interface (like `:class:`bytearray``).

In all but the last case, behaviour is as for the `:func:`~fcntl.fcntl`` function.

If a mutable buffer is passed, then the behaviour is determined by the value of the `*mutate_flag*` parameter.

If it is false, the buffer's mutability is ignored and behaviour is as for a read-only buffer, except that the 1024 byte limit mentioned above is avoided -- so long as the buffer you pass is at least as long as what the operating system wants to put there, things should work.

If `*mutate_flag*` is true (the default), then the buffer is (in effect) passed to the underlying `:func:`ioctl`` system call, the latter's return code is passed back to the calling Python, and the buffer's new contents reflect the action of the `:func:`ioctl``. This is a slight simplification, because if the supplied buffer is less than 1024 bytes long it is first copied into a static buffer 1024 bytes long which is then passed to `:func:`ioctl`` and copied back into the supplied buffer.

If the `:c:func:`ioctl`` fails, an `:exc:`OSError`` exception is raised.

An example::

```
>>> import array, fcntl, struct, termios, os
>>> os.getpgrp()
13341
>>> struct.unpack('h', fcntl.ioctl(0, termios.TIOCGPGRP, " "))[0]
13341
>>> buf = array.array('h', [0])
>>> fcntl.ioctl(0, termios.TIOCGPGRP, buf, 1)
0
>>> buf
array('h', [13341])

.. audit-event:: fcntl.ioctl fd,request,arg fcntl.ioctl
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-

main\Doc\library\ (cpython-main) (Doc) (library)fcntl.rst, line 130)

Unknown directive type "function".

```
.. function:: flock(fd, operation)
```

Perform the lock operation **operation** on file descriptor **fd** (file objects providing a `:meth:`~io.IOBase.fileno`` method are accepted as well). See the Unix manual `:manpage:`flock(2)`` for details. (On some systems, this function is emulated using `:c:func:`fcntl``.)

If the `:c:func:`flock`` fails, an `:exc:`OSError`` exception is raised.

```
.. audit-event:: fcntl.flock fd,operation fcntl.flock
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library)fcntl.rst, line 142)

Unknown directive type "function".

```
.. function:: lockf(fd, cmd, len=0, start=0, whence=0)
```

This is essentially a wrapper around the `:func:`~fcntl.fcntl`` locking calls. **fd** is the file descriptor (file objects providing a `:meth:`~io.IOBase.fileno`` method are accepted as well) of the file to lock or unlock, and **cmd** is one of the following values:

```
* :const:`LOCK_UN` -- unlock
* :const:`LOCK_SH` -- acquire a shared lock
* :const:`LOCK_EX` -- acquire an exclusive lock
```

When **cmd** is `:const:`LOCK_SH`` or `:const:`LOCK_EX``, it can also be bitwise ORed with `:const:`LOCK_NB`` to avoid blocking on lock acquisition. If `:const:`LOCK_NB`` is used and the lock cannot be acquired, an `:exc:`OSError`` will be raised and the exception will have an **errno** attribute set to `:const:`EACCES`` or `:const:`EAGAIN`` (depending on the operating system; for portability, check for both values). On at least some systems, `:const:`LOCK_EX`` can only be used if the file descriptor refers to a file opened for writing.

len is the number of bytes to lock, **start** is the byte offset at which the lock starts, relative to **whence**, and **whence** is as with `:func:`io.IOBase.seek``, specifically:

```
* :const:`0` -- relative to the start of the file (:data:`os.SEEK_SET`)
* :const:`1` -- relative to the current buffer position (:data:`os.SEEK_CUR`)
* :const:`2` -- relative to the end of the file (:data:`os.SEEK_END`)
```

The default for **start** is 0, which means to start at the beginning of the file. The default for **len** is 0 which means to lock to the end of the file. The default for **whence** is also 0.

```
.. audit-event:: fcntl.lockf fd,cmd,len,start,whence fcntl.lockf
```

Examples (all on a SVR4 compliant system):

```
import struct, fcntl, os

f = open(...)
rv = fcntl.fcntl(f, fcntl.F_SETFL, os.O_NDELAY)

lockdata = struct.pack('hllhh', fcntl.F_WRLCK, 0, 0, 0, 0)
rv = fcntl.fcntl(f, fcntl.F_SETLKW, lockdata)
```

Note that in the first example the return value variable *rv* will hold an integer value; in the second example it will hold a `:class:`bytes`` object. The structure lay-out for the *lockdata* variable is system dependent --- therefore using the `:func:`flock`` call may be better.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library)fcntl.rst, line 186); [backlink](#)

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library)fcntl.rst, line 186); [backlink](#)

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) fcntl.rst, line 192)

Unknown directive type "seealso".

.. seealso::

Module :mod:`os`

If the locking flags :data:`~os.O_SHLOCK` and :data:`~os.O_EXLOCK` are present in the :mod:`os` module (on BSD only), the :func:`os.open` function provides an alternative to the :func:`lockf` and :func:`flock` functions.