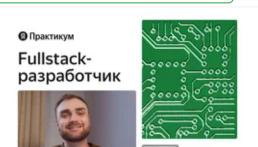
13.09.2023, 22:58 Модуль fcntl в Python

ХОЧУ ПОМОЧЬ ПРОЕКТУ

Модуль fcntl в Python

Сообщить об ошибке.



n practicum.yandex.ru

РЕКЛАМА • 16+

Станьте Fullstack-разработчиком на курсе от Яндекса.

16 проектов в портфолио • Поддержка 24/7 • 480 часов практики • Коммерческие проекты

Узнать больше

/ Модуль fcntl в Python



водом файловых дескрипторов

ение файлами и вводом/выводом файловых дескрипторов. Это интерфейс для базовых) и ioctl(). Полное описание этих системных вызовов можно посмотреть командами man

едоступен на платформах WebAssembly wasm32-emscripten и wasm32-wasi.

Bce функции модуля fcntl, в качестве первого аргумента, принимают файловый дескриптор fd. Это может быть целочисленный файловый дескриптор, например, возвращаемый функцией sys.stdin.fileno(), или объект io.IOBase, такой как сам sys.stdin, который предоставляет fileno(), возвращающий подлинный файловый дескриптор.

<u>Изменено в Python 3.9</u>: в macOS модуль fcntl предоставляет константу fcntl.F_GETPATH, которая получает путь к файлу из файлового дескриптора. В Linux (=3.15) модуль fcntl предоставляет константы fcntl.F_OFD_GETLK, fcntl.F_OFD_SETLK и fcntl.F_OFD_SETLKW, которые используются при работе с блокировками описания открытых файлов.

<u>Изменено в Python 3.10</u>: В Linux = 2.6.11 модуль fcntl предоставляет константы fcntl.F_GETPIPE_SZ и fcntl.F_SETPIPE_SZ, которые позволяют проверять и изменять размер канала соответственно.

<u>Изменено в Python 3.11</u>: В FreeBSD модуль fcntl предоставляет константы fcntl.F_DUP2FD и fcntl.F_DUP2FD_CLOEXEC, которые позволяют дублировать файловый дескриптор, последний дополнительно устанавливает флаг fcntl.FD_CLOEXEC.

Модуль fcntl определяет следующие функции:

- fcntl.fcntl() выполняет операцию cmd над файловым дескриптором fd;
- fcntl.ioctl() <u>идентична fcntl.fcntl(), но обработка аргументов сложнее</u>;
- fcntl.flock() выполняет операцию блокировки над файловым дескриптором;
- fcntl.lockf() <u>обертка для блокирующих вызовов fcntl.fcntl()</u>;
- Примеры использования.

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

Функция fcntl.fcntl() выполняет операцию cmd над файловым дескриптором fd (также принимаются файловые объекты, предоставляющие метод fileno()).

Значения, используемые для аргумента cmd, зависят от операционной системы и доступны как константы модуля fcntl с теми же именами, что и в соответствующих заголовочных файлах языка С (можно посмотреть в Unix командами man fcntl).

Аргумент arg может быть либо целым числом, либо байтовым объектом.

- Если arg целое число, то возвращаемое значение этой функции является целочисленным значением, возвращаемым системным вызовом fcntl().
- Если arg являются байты, то возвращаемое значение представляет собой двоичную структуру, например, созданный struct.pack(). Двоичные данные копируются в буфер, адрес которого передается системному вызову fcntl(). Возвращаемое значение после успешного вызова это содержимое буфера, преобразованное в байтовый объект. Длина возвращаемого объекта будет такой же, как длина аргумента arg (ограничено 1024 байтами). Если информация, возвращаемая операционной системой в буфере, превышает 1024 байта, это, скорее всего, приведет к нарушению Вверх нтации или более тонкому повреждению данных.

Если fcntl.fcntl() не работает, то возникает исключение OSError.

Вызывает событие аудита fcntl.fcntl c аргументами fd, cmd, arg.

```
fcpetal institute [(fd, request, arg=0, mutate_flag=True):
```

Функция fcntl.ioctl() идентична функции fcntl.fcntl(), за исключением того, что обработка аргументов еще сложнее.

Аргумент request ограничен значениями, которые могут уместиться в 32 бита. Дополнительные константы, представляющие интерес для использования в качестве аргумента request, можно найти в модуле стандартной библиотекеtermios под теми же именами, что и в соответствующих заголовочных файлах языка С.

Аргумент arg может быть целым числом, объектом, поддерживающим интерфейс буфера только для чтения (например, bytes), или объектом, поддерживающим интерфейс буфера чтения-записи (например, bytearray).

Во всех случаях, кроме последнего, поведение такое же, как и для функции fcntl.fcntl().

Если передается изменяемый буфер, то поведение определяется значением аргументом mutate_flag.

Ecли apryment mutate_flag=False, то изменчивость буфера игнорируется, и поведение такое же, как и для буфера только для чтения, за исключением того, что упомянутое выше ограничение в 1024 байта избегается до тех пор, пока буфер, который передается, имеет по крайней мере длину, которую операционная система хочет поместить туда.

Если mutate_flag=True (по умолчанию), то буфер (по сути) передается базовому системному вызову ioctl(), код возврата последнего передается обратно вызывающему Python, а новое содержимое буфера отражает действие ioctl(). Это небольшое упрощение, т.к. если предоставленный буфер имеет длину менее 1024 байт, он сначала копируется в статический буфер длиной 1024 байта, который затем передается в ioctl() и копируется обратно в предоставленный буфер.

В случае сбоя fcntl.ioctl() возникает исключение OSError.

Пример:

```
>>> 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])
```

fcntl.flock(fd, operation):

Функция fcntl.flock() выполняет операцию блокировки над файловым дескриптором fd (также принимаются файловые объекты, предоставляющие метод .fileno()). Подробности смотрите в руководстве по Unix (команда man flock) flock(2).

В некоторых системах эта функция эмулируется с помощью <u>fcntl.fcntl()</u>.

В случае сбоя fcntl.flock() возникает <u>исключение OSError</u>.

fcnt1.lockf(fd, cmd, len=0, start=0, whence=0):

Функция fcntl.lockf() по сути, обертка для блокирующих вызовов fcntl.fcntl().

Аргумент fd - это файловый дескриптор файла для блокировки или разблокировки (также принимаются файловые объекты, с методом .fileno()),

Аргумент cmd - одно из следующих значений:

- fcntl.LOCK_UN разблокировать
- fcntl.LOCK_SH получить общую блокировку
- fcntl.LOCK_EX получить эксклюзивную блокировку

Brenx

13.09.2023, 22:58 Модуль fcntl в Python

Когда cmd имеет значение LOCK_SH или LOCK_EX, то к нему также можно применить побитовое ИЛИ с LOCK_NB, чтобы избежать блокировки при получении блокировки. Если используется LOCK_NB и блокировка не может быть получена, то будет вызвано сообщение об ошибке OSError, а атрибуту исключения будет присвоено значение EACCES или EAGAIN (в зависимости от операционной системы; для переносимости необходимо проверять оба значения). По крайней мере, в некоторых системах LOCK_EX может использоваться только в том случае, если дескриптор файла ссылается на файл, открытый для записи.

Аргумент len - количество байт для блокировки. Аргументstart- смещение в байтах, с которого начинается блокировка, относительноwhence`.

Аргумент whence имеет значения как и в случае с io.IOBase.seek(), а именно:

- 0 относительно начала файла (os.SEEK_SET)
- 1 относительно текущей позиции буфера (os.SEEK_CUR)
- 2 относительно конца файла (os.SEEK_END)

Значение по умолчанию для start равно 0, что означает запуск в начале файла. Значение по умолчанию для len равно 0, что означает блокировку до конца файла. Значение по умолчанию для параметра whence также равно 0.

Примеры использования:

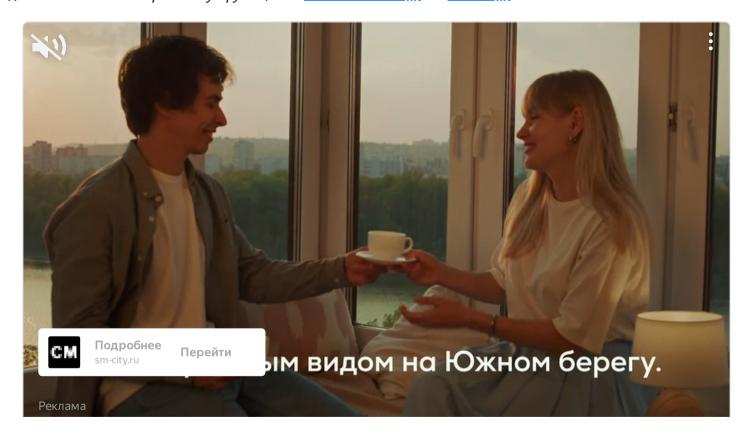
```
import struct, fcntl, os

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

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

<u>Обратите внимание</u>, что в первом случае переменная возвращаемого значения rv будет содержать целочисленное значение; во втором примере она будет содержать объект bytes. Структура переменной lockdata зависит от системы, поэтому лучше использовать вызов <u>fcntl.flock()</u>.

Если в [модуле os присутствуют флаги блокировки O_SHLOCK и O_EXLOCK (только в BSD), то функция os.open() предоставляет альтернативу функциям fcntl.lockf() и flock().



DOCS-Python.ru™, 2023 г.

(Внимание! При копировании материала ссылка на источник обязательна)

@docs_python_ru