Модуль selectors в Python, готовность файлов к вводу/выводу

Ув ости ввода/вывода нескольких файловых объектов

Мо высокоуровневое и эффективное мультиплексирование ввода-вывода, основанное на пр и пользователям не нужно точный контроль над примитивами операционной системы, то ре от модуль вместо модуля select.

Мо есколько конкретных реализаций: selectors.EpollSelector(), selectors.SelectSelector() и зовать для ожидания уведомления о готовности ввода-вывода для нескольких файловых об ект" относится к любому объекту с методом file.fileno() или необработанным файловым де

Кл or - это псевдоним наиболее эффективной реализации, доступной на текущей платформе: эт м по умолчанию для большинства пользователей.

Пр х файловых объектов зависит от платформы: в Windows поддерживаются работа только с сокетами, но не с каналами, тогда как в Unix поддерживаются оба варианта. Также могут поддерживаться некоторые другие типы, например FIFO или специальные файловые устройства.

События представляют собой побитовую маску, указывающую, какие события ввода-вывода следует ожидать для данного файлового объекта. Это может быть комбинация констант модуля:

- selectors.EVENT_READ - доступно для чтения;
- selectors.EVENT_WRITE - доступно для записи;

Классы определяемые модулем.

Иерархия классов модуля:

```
BaseSelector
+-- SelectSelector
+-- PollSelector
+-- EpollSelector
+-- DevpollSelector
+-- KqueueSelector
```

Абстрактный базовый класс selectors.BaseSelector и все его конкретные реализации поддерживают протокол диспетчера контекста.

selectors.SelectorKey():

Класс selectors.SelectorKey() - это именованное множество, используемое для связывания файлового объекта с его базовым файловым дескриптором, выбранной маской события и вложенными данными. Он возвращается несколькими методами BaseSelector.

Класс selectors.SelectorKey() следующие атрибуты:

- SelectorKey.fileobj - зарегистрированный файловый объект.
- SelectorKey.fd - базовый файловый дескриптор.
- SelectorKey.events - события, которые необходимо дождаться для этого файлового объекта.

- SelectorKey.data - необязательные непрозрачные данные, связанные с этим файловым объектом: например, они могут использоваться для хранения идентификатора сеанса для каждого клиента.

selectors.DefaultSelector():

Класс selectors.DefaultSelector(), автоматически выбирает и использует наиболее эффективную реализацию, доступную на текущей платформе. Этот класс должен быть выбором по умолчанию для большинства пользователей.

Наследует методы от абстрактного класса |selectors.BaseSelector()|.

selectors.SelectSelector():

Класс selectors.DefaultSelector() создан на основе базового объекта select.select().

Наследует методы от абстрактного базового класса selectors.BaseSelector.

selectors.PollSelector():

Класс selectors.PollSelector() создан на основе базового объекта select.poll().

Наследует методы от абстрактного базового класса selectors.BaseSelector.

selectors.EpollSelector():

Класс selectors.EpollSelector() создан на основе базового объекта select.epoll().

Наследует методы от абстрактного базового класса selectors.BaseSelector и определяет один дополнительный метод:

- EpollSelector.fileno() - возвращает файловый дескриптор, используемый базовым объектом select.epoll().

Пример неблокирующего сокет сервера с использованием модуля selectors.

Вот простой эхо сервер.

```
import selectors
import socket

sel = selectors.DefaultSelector()

def accept(sock, mask):
    # Должен быть готов к чтению
    conn, addr = sock.accept()
    print('Создан сокет:', conn, 'для адреса:', addr)
    # включаем для него неблокирующий режим
    conn.setblocking(False)
    # регистрируем для прослушивания событий клиента
    sel.register(conn, selectors.EVENT_READ, read)

def read(conn, mask):
    # Должен быть готов к чтению
    data = conn.recv(1000)
    if data:
        print('Отвечаем: ', repr(data), 'сокету =>', conn)
        conn.send(data)
    else:
        print('Закрытие соединения', conn)
        sel.unregister(conn)
        conn.close()

sock = socket.socket()
sock.bind(('localhost', 8008))
sock.listen(5)
sock.setblocking(False)
sel.register(sock, selectors.EVENT_READ, accept)

while True:
    events = sel.select()
    for key, mask in events:
```

```
        callback = key.data
        callback(key.fileobj, mask)
```


Содержание раздела: