
Descritor de arquivo de temporizador

Release 3.13.2

Guido van Rossum and the Python development team

fevereiro 17, 2025

Python Software Foundation
Email: docs@python.org

Sumário

1 Exemplos

1

Versão

1.13

Este documento discute o suporte do Python para o descritor de arquivo de temporizador do Linux.

1 Exemplos

O exemplo a seguir mostra como usar um descritor de arquivo de temporizador para executar uma função duas vezes por segundo:

```
# Scripts práticos devem realmente usar um timer sem bloqueio.
# Usamos um timer com bloqueio aqui para simplificar.
import os, time

# Cria um descritor de arquivo de temporizador
fd = os.timerfd_create(time.CLOCK_REALTIME)

# Inicia o temporizador em 1 segundo, com um intervalo de meio segundo
os.timerfd_settime(fd, initial=1, interval=0.5)

try:
    # Processa eventos do temporizador quatro vezes.
    for _ in range(4):
        # read() ser bloqueado até o temporizador expirar
        _ = os.read(fd, 8)
        print("Timer expired")
finally:
    # Lembre-se de fechar o descritor de arquivo de temporizador!
    os.close(fd)
```

Para evitar a perda de precisão causada pelo tipo `float`, os descritores de arquivos de temporizador permitem especificar a expiração inicial e o intervalo em nanossegundos inteiros com variantes `_ns` das funções.

Este exemplo mostra como `epoll()` pode ser usado com descritores de arquivo de temporizador para esperar até que o descritor de arquivo esteja pronto para leitura:

```
import os, time, select, socket, sys

# Cria um objeto epoll
ep = select.epoll()

# Neste exemplo, usa o endereço de loopback para enviar o comando "stop" ao
↳servidor.
#
# $ telnet 127.0.0.1 1234
# Trying 127.0.0.1...
# Connected to 127.0.0.1.
# Escape character is '^]'.
# stop
# Connection closed by foreign host.
#
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.bind(("127.0.0.1", 1234))
sock.setblocking(False)
sock.listen(1)
ep.register(sock, select.EPOLLIN)

# Cria descritores de arquivo de temporizador no modo não bloqueante.
num = 3
fds = []
for _ in range(num):
    fd = os.timerfd_create(time.CLOCK_REALTIME, flags=os.TFD_NONBLOCK)
    fds.append(fd)
    # Registra o descritor de arquivo do temporizador para eventos de leitura
    ep.register(fd, select.EPOLLIN)

# Inicia o temporizador com os.timerfd_settime_ns() em nanossegundos.
# O temporizador 1 dispara a cada 0,25 segundos; o timer 2 a cada 0,5 segundos;
↳etc.
for i, fd in enumerate(fds, start=1):
    one_sec_in_nsec = 10**9
    i = i * one_sec_in_nsec
    os.timerfd_settime_ns(fd, initial=i//4, interval=i//4)

timeout = 3
try:
    conn = None
    is_active = True
    while is_active:
        # Aguarda o temporizador expirar por 3 segundos.
        # epoll.poll() retorna uma lista de pares (fd, event).
        # fd é um descritor de arquivo.
        # sock e conn[=valor retornado de socket.accept()] são objetos de socket,
↳não descritores de arquivo.
        # Então use sock.fileno() e conn.fileno() para obter os descritores de
↳arquivo.
        events = ep.poll(timeout)

        # Se mais de um descritor de arquivo de timer estiver pronto para leitura
↳ao mesmo tempo,
        # epoll.poll() retorna uma lista de pares (fd, event).
```

(continua na próxima página)

```

#
# Neste exemplo de configurações,
#   O 1º temporizador dispara a cada 0,25 segundos em 0,25 segundos. (0,
↪25, 0,5, 0,75, 1,0, ...)
#   O 2º temporizador dispara a cada 0,5 segundos em 0,5 segundos. (0,5,
↪1,0, 1,5, 2,0, ...)
#   O 3º temporizador dispara a cada 0,75 segundos em 0,75 segundos. (0,
↪75, 1,5, 2,25, 3,0, ...)
#
#   Em 0,25 segundos, apenas o 1º temporizador dispara.
#   Em 0,5 segundos, o 1º temporizador e o 2º temporizador disparam ao
↪mesmo tempo.
#   Em 0,75 segundos, o 1º temporizador e o 3º temporizador disparam ao
↪mesmo tempo.
#   Em 1,5 segundos, o 1º temporizador, o 2º temporizador e o 3º
↪temporizador disparam ao mesmo tempo.
#
# Se um descritor de arquivo de temporizador for sinalizado mais de uma
↪vez desde
# a última chamada de os.read(), os.read() retornará o número de
↪sinalizados
# como ordem de host de bytes de classe.
print(f"Signaled events={events}")
for fd, event in events:
    if event & select.EPOLLIN:
        if fd == sock.fileno():
            # Verifica se há uma requisição de conexão.
            print(f"Accepting connection {fd}")
            conn, addr = sock.accept()
            conn.setblocking(False)
            print(f"Accepted connection {conn} from {addr}")
            ep.register(conn, select.EPOLLIN)
        elif conn and fd == conn.fileno():
            # Verifica se há dados para ler.
            print(f"Reading data {fd}")
            data = conn.recv(1024)
            if data:
                # Você deveria capturar exceção UnicodeDecodeError por
↪segurança.

                cmd = data.decode()
                if cmd.startswith("stop"):
                    print(f"Stopping server")
                    is_active = False
                else:
                    print(f"Unknown command: {cmd}")
            else:
                # Acabaram os dados, fecha a conexão
                print(f"Closing connection {fd}")
                ep.unregister(conn)
                conn.close()
                conn = None
        elif fd in fds:
            print(f"Reading timer {fd}")
            count = int.from_bytes(os.read(fd, 8), byteorder=sys.byteorder)
            print(f"Timer {fds.index(fd) + 1} expired {count} times")
        else:

```

```

        print(f"Unknown file descriptor {fd}")
finally:
    for fd in fds:
        ep.unregister(fd)
        os.close(fd)
    ep.close()

```

Este exemplo mostra como `select()` pode ser usado com descritores de arquivo de temporizador para esperar até que o descritor de arquivo esteja pronto para leitura:

```

import os, time, select, socket, sys

# Neste exemplo, usa o endereço de loopback para enviar o comando "stop" ao
# servidor.
#
# $ telnet 127.0.0.1 1234
# Trying 127.0.0.1...
# Connected to 127.0.0.1.
# Escape character is '^]'.
# stop
# Connection closed by foreign host.
#
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.bind(("127.0.0.1", 1234))
sock.setblocking(False)
sock.listen(1)

# Cria descritores de arquivo de temporizador no modo não bloqueante.
num = 3
fds = [os.timerfd_create(time.CLOCK_REALTIME, flags=os.TFD_NONBLOCK)
        for _ in range(num)]
select_fds = fds + [sock]

# Inicia o temporizador com os.timerfd_settime() em segundos.
# O temporizador 1 dispara a cada 0,25 segundos; o timer 2 a cada 0,5 segundos;
# etc.
for i, fd in enumerate(fds, start=1):
    os.timerfd_settime(fd, initial=i/4, interval=i/4)

timeout = 3
try:
    conn = None
    is_active = True
    while is_active:
        # Aguarda o temporizador expirar por 3 segundos.
        # select.select() retorna uma lista de objetos ou descritores de arquivos.
        rfd, wfd, xfd = select.select(select_fds, select_fds, select_fds, timeout)
        for fd in rfd:
            if fd == sock:
                # Verifica se há uma requisição de conexão.
                print(f"Accepting connection {fd}")
                conn, addr = sock.accept()
                conn.setblocking(False)
                print(f"Accepted connection {conn} from {addr}")
                select_fds.append(conn)
            elif conn and fd == conn:

```

(continua na próxima página)

```

        # Verifica se há dados para ler.
        print(f"Reading data {fd}")
        data = conn.recv(1024)
        if data:
            # Você deveria capturar exceção UnicodeDecodeError por
↪segurança.

            cmd = data.decode()
            if cmd.startswith("stop"):
                print(f"Stopping server")
                is_active = False
            else:
                print(f"Unknown command: {cmd}")
        else:
            # Acabaram os dados, fecha a conexão
            print(f"Closing connection {fd}")
            select_fds.remove(conn)
            conn.close()
            conn = None
    elif fd in fds:
        print(f"Reading timer {fd}")
        count = int.from_bytes(os.read(fd, 8), byteorder=sys.byteorder)
        print(f"Timer {fds.index(fd) + 1} expired {count} times")
    else:
        print(f"Unknown file descriptor {fd}")
finally:
    for fd in fds:
        os.close(fd)
    sock.close()
    sock = None

```