
計時器檔案描述器 (timerfd) 指南

發行 3.14.0rc3

Guido van Rossum and the Python development team

10 月 01, 2025

Python Software Foundation
Email: docs@python.org

Contents

1 範例	1
------	---

發行版本
1.13

此篇指南探討 Python 對 Linux 計時器檔案描述器的支援。

1 范例

以下範例顯示如何使用計時器檔案描述器來每秒執行函式兩次：

```
# 實際本應該使用非阻塞計時器，
# 在這簡化了簡單起見我們使用阻塞計時器。
import os, time

# 建立計時器檔案描述器
fd = os.timerfd_create(time.CLOCK_REALTIME)

# 在 1 秒後開始計時器，間隔半秒
os.timerfd_settime(fd, initial=1, interval=0.5)

try:
    # 處理計時器事件四次。
    for _ in range(4):
        # read() 會阻塞直到計時器到期
        _ = os.read(fd, 8)
        print("Timer expired")
finally:
    # 記得關閉計時器檔案描述器！
    os.close(fd)
```

為了避免由 float 型造成的精度損失，計時器檔案描述器允許使用函式的 `_ns` 變體以整數奈秒單位指定初始到期時間和間隔。

此範例展示如何將 `epoll()` 與計時器檔案描述器一起使用，用於等待檔案描述器直到它準備好讀取：

```

import os, time, select, socket, sys

# 建立 epoll 物件
ep = select.epoll()

# 在此範例中，使用回送位址 (loopback address) 向伺服器發送 "stop" 命令。
#
# $ 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)

# 以非阻塞模式建立計時器檔案描述器。
num = 3
fds = []
for _ in range(num):
    fd = os.timerfd_create(time.CLOCK_REALTIME, flags=os.TFD_NONBLOCK)
    fds.append(fd)
    # 讀取事件 計時器檔案描述器
    ep.register(fd, select.EPOLLIN)

# 使用 os.timerfd_settime_ns() 以奈秒單位動計時器。
# 計時器 1 每 0.25 秒觸發一次；計時器 2 每 0.5 秒觸發一次；以此類推
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:
        # 等待計時器在 3 秒到期。
        # epoll.poll() 回傳一個 (fd, event) 配對的串列。
        # fd 是檔案描述器。
        # sock 和 conn[=socket.accept() 的回傳值] 是 socket 物件，不是檔案描述器。
        # 所以要使用 sock.fileno() 和 conn.fileno() 來取得檔案描述器。
        events = ep.poll(timeout)

        # 如果同時有多個計時器檔案描述器準備好讀取，
        # epoll.poll() 會回傳一個 (fd, event) 配對的串列。
        #
        # 在此範例設定中，
        # 第 1 個計時器在 0.25 秒每 0.25 秒觸發一次。(0.25, 0.5, 0.75, 1.0, ...)
        # 第 2 個計時器在 0.5 秒每 0.5 秒觸發一次。(0.5, 1.0, 1.5, 2.0, ...)
        # 第 3 個計時器在 0.75 秒每 0.75 秒觸發一次。(0.75, 1.5, 2.25, 3.0, ...)
        #
        # 在 0.25 秒時，只有第 1 個計時器觸發。
        # 在 0.5 秒時，第 1 個計時器和第 2 個計時器同時觸發。
        # 在 0.75 秒時，第 1 個計時器和第 3 個計時器同時觸發。
        # 在 1.5 秒時，第 1、2、3 個計時器同時觸發。
        #
        # 如果計時器檔案描述器自上次 os.read() 呼叫以來被觸發多次，

```

(繼續下一頁)

```

# os.read() 會以主機類位元組順序回傳被觸發次數。
print(f"Signaled events={events}")
for fd, event in events:
    if event & select.EPOLLIN:
        if fd == sock.fileno():
            # 檢查是否有連請求。
            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():
            # 檢查是否有資料要讀取。
            print(f"Reading data {fd}")
            data = conn.recv(1024)
            if data:
                # 了安全起見，你應該捕獲 UnicodeDecodeError 例外。
                cmd = data.decode()
                if cmd.startswith("stop"):
                    print(f"Stopping server")
                    is_active = False
                else:
                    print(f"Unknown command: {cmd}")
            else:
                # 有更多資料，關閉連
                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()

```

此範例展示如何將 `select()` 與計時器檔案描述器一起使用，用於等待檔案描述器直到它準備好讀取：

```

import os, time, select, socket, sys

# 在此範例中，使用回送位址向伺服器發送 "stop" 命令。
#
# $ 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)

# 以非阻塞模式建立計時器檔案描述器。
num = 3

```

```

fds = [os.timerfd_create(time.CLOCK_REALTIME, flags=os.TFD_NONBLOCK)
        for _ in range(num)]
select_fds = fds + [sock]

# 使用 os.timerfd_settime() 以秒為單位啟動計時器。
# 計時器 1 每 0.25 秒觸發一次；計時器 2 每 0.5 秒觸發一次；以此類推
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:
        # 等待計時器在 3 秒內到期。
        # select.select() 回傳檔案描述器或物件的串列。
        rfd, wfd, xfd = select.select(select_fds, select_fds, select_fds, timeout)
        for fd in rfd:
            if fd == sock:
                # 檢查是否有連接請求。
                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:
                # 檢查是否有資料要讀取。
                print(f"Reading data {fd}")
                data = conn.recv(1024)
                if data:
                    # 為了安全起見，你應該捕獲 UnicodeDecodeError 例外。
                    cmd = data.decode()
                    if cmd.startswith("stop"):
                        print(f"Stopping server")
                        is_active = False
                    else:
                        print(f"Unknown command: {cmd}")
                else:
                    # 沒有更多資料，關閉連接
                    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

```