# MỤC LỤC

# Bài 8: Lập trình Socket

**Mục tiêu:**

- *Hiểu và giải thích được các hàm cơ bản trong lập trình socket sử dụng ngôn ngữ python*
- *Lập trình một ứng dụng đơn giản minh họa mô hình kết nối client-server*

**Nội dung chính:**

- *Các bước thiết lập kết nối socket theo giao thức TCP client-server bằng ngôn ngữ python*
- *Ví dụ minh họa về kết nối socket*
- *Xây dựng ứng dụng đơn giản theo mô hình client-server*

Socket là điểm đầu/cuối (endpoint) trong một kênh giao tiếp. Các socket có thể giao tiếp bên trong một tiến trình, giữa các tiến trình trong cùng một thiết bị, hoặc giữa các tiến trình khác nhau.

## Socket module trong python

Khởi tạo một socket, bạn phải sử dụng hàm:

```
s = socket.socket(family, type, protocol=0)
```

- **Family**: giao thức được sử dụng trong kỹ thuật vận chuyển. Có thể là 1 trong 2 hằng số: AF_UNIX hoặc **AF_INET**
- **Type**: **SOCK_STREAM** cho giao thức hướng kết nối, hoặc **SOCK_DGRAM** cho giao thức hướng không kết nối
- **Protocol**: dùng để xác định số lượng giao thức trong cùng một domain. Thường được để trống, mặc định là 0

Khi đã khởi tạo đối tượng socket, bạn có thể sử dụng các phương thức để tạo chương trình cho Client hoặc Server:

### Các phương thức sử dụng cho Server socket

| Phương thức | Miêu tả |
|---|---|
| s.bind((addr,port)) | Gắn kết địa chỉ (hostname, port number) tới socket |
| s.listen(backlog) | Thiết lập và bắt đầu lắng TCP listenter để lắng nghe yêu cầu kết nối từ client<br>backlog: số lượng tối đa các kết nối có thể có trong hàng đợi (0-5) |
| s.accept() | Chấp nhận một cách thụ động kết nối TCP client. Đợi cho tới khi có kết nối tới thì chấp nhận |

### Các phương thức sử dụng cho Client socket

| Phương thức | Miêu tả |
|---|---|
| s.connect((addr,port)) | Khởi tạo kết nối TCP server |

### Các phương thức chung cho Socket

| Phương thức | Miêu tả |
|---|---|
| s.**recv**(buflen[,flags]) | Nhận dữ liệu từ socket (**TCP** msg)<br>buflen: số bytes tối đa có thể nhận<br>Giá trị trả về là một chuỗi thể hiện dữ liệu nhận được |
| s.**send**(data[,flag]) | Truyền dữ liệu qua socket (**TCP** msg)<br>Trả về số lượng byte đã được truyền |
| s.**recvfrom**(buflen[,flags]) | Nhận **UDP** message<br>Receive data from the socket, up to buflen bytes, returning also the remote host and port from which the data came |
| s.**sendto**(data[,flag],addr) | Truyền **UDP** message<br>Send the *data* through the socket<br>Return the number of bytes sent |
| s.sendall(data[,flag]) | Send *data* to the socket (this method continues to send data from string until either all data has been sent or an error occurs) *None* is returned on success |
| s.close() | Đóng socket |
| s.gethostname() | Trả về hostname |

**Thí dụ 1: Viết một chương trình TCPServer đơn giản**

Đầu tiên, bạn tạo một đối tượng socket.

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

Tham số **AF_INET cho biết chúng ta sử dụng IPv4, SOCK_STREAM là dùng giao thức TCP**. Ngoài ra, còn một số giá trị khác như **AF_INET6 là dùng IPv6, AF_UNIX là chỉ kết nối các ứng dụng trong một máy** (không dùng mạng), **SOCK_DGRAM là dùng giao thức UDP.**

Sau đó, đối tượng socket được sử dụng để gọi các hàm khác để thiết lập một socket server:

Phương thức bind(hostname, port) chỉ định socket lắng nghe với địa chỉ IP của máy lấy từ phương thức gethostname() trên port xác định.

Phương thức listen() cho python biết socket này là một server, tham số của phuong thức này là số lượng các kết nối có thể có trong hàng đợi

Phương thức accept() đưa server vào trạng thái chờ đợi cho đến khi có một client kết nối tới port mà bạn đã xác định, phương thức này trả về một đối tượng connection biểu diễn kết nối tới client đó.

```
#SimpleServer.py
import socket

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
host = socket.gethostname()
port = 12345
s.bind((host, port))
s.listen(5)
while True:
    c, addr = s.accept()
    print("Da chap nhan ket noi tu ", addr)
    output = "Cam on ban da ket noi"
    c.sendall(output.encode("utf-8"))
    c.close()
```

**Thí dụ 2: Viết một chương trình Client đơn giản**

Hàm socket.connect(hostname, port) mở một kết nối TCP tới hostname trên port đã cho. Khi có một socket đã được mở, bạn có thể đọc thông tin từ nó giống như bất kỳ đối tượng IO nào

```
#SimpleClient
import socket

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
host = socket.gethostname()
port = 12345
```

```
s.connect((host, port))
print(s.recv(1024))
s.close()
```

# BÀI TẬP CHƯƠNG 8

**Bài 1**: Từ thí dụ trên, sinh viên viết 1 echo-server theo yêu cầu sau:

*- Server: lắng nghe trên một port xác định (do sinh viên quy định), chấp nhận yêu cầu kết nối từ một client, hiển thị thông tin client, nhận một chuỗi từ client sau đó chuyển trả lại cho client*

*- Client: kết nối tới server thông qua một port xác định (do sinh viên quy định), gởi chuỗi đến server, nhận chuỗi mới từ server và hiển thị trên màn hình của client*

**Bài 2**: *Từ thí dụ trên, sinh viên viết lại hệ thống một client-server theo yêu cầu sau:*

*- Server: lắng nghe trên một port xác định (do sinh viên quy định), chấp nhận yêu cầu kết nối từ một client, hiển thị thông tin client, nhận một chuỗi từ client sau đó chuyển thành chữ hoa và trả chuỗi mới về cho client*

*- Client: kết nối tới server thông qua một port xác định (do sinh viên quy định), gởi chuỗi đến server, nhận chuỗi mới từ server và hiển thị trên màn hình của client*

# TÀI LIỆU THAM KHẢO

https://daynhauhoc.com/t/lam-server-python-truy-cap-duoc-tren-internet/51955/4

https://www.academia.edu/33214759/L%E1%BA%ADp_tr%C3%ACnh_m%E1%BA%A1ng_trong_Python

https://www.youtube.com/watch?v=O8bIjCFhgPg