وزارت علوم، تحقیقات و فناوری
موسسه آموزش عالی زند شیراز

تمرین

اینترنت اشیا

استاد

محمد زارع

به کوشش

مهدی جوانشیر

تاریخ

آبان ۱۴۰۳

یک برنامه ساده برای ارتباط بین دو دستگاه در حالت چت کردن با استفاده از پروتکل TCP/IP بنویسید.

هدف

ما می‌خواهیم یک برنامه چت ساده بین دو دستگاه با استفاده از پروتکل TCP/IP بنویسیم. این برنامه از دو بخش اصلی تشکیل می‌شود :**سرور** و **کلاینت** .سرور مسئول گوش دادن به درخواست‌های ورودی و پاسخ دادن به پیام‌های کلاینت است. کلاینت به سرور متصل شده و پیام‌های خود را ارسال و پیام‌های دریافتی از سرور را نمایش می‌دهد. این ارتباط به صورت **ساده و تعاملی** پیاده‌سازی می‌شود، به طوری که هر دو طرف (سرور و کلاینت) می‌توانند به تبادل پیام بپردازند.

نحوه کار پروتکلTCP/IP

TCP/IP (Transmission Control Protocol/Internet Protocol)پروتکلی است که برای ارتباط امن و قابل اعتماد بین دستگاه‌ها در شبکه استفاده می‌شود. در این نوع ارتباط، داده‌ها به صورت پیوسته و بدون از دست رفتن انتقال داده می‌شوند.

پروتکل TCP ارتباط را از طریق ایجاد یک اتصال امن برقرار می‌کند. ابتدا کلاینت به سرور متصل می‌شود و یک "ارتباط پایدار" بین آن‌ها ایجاد می‌شود. این اتصال تا زمانی که یکی از طرفین ارتباط را خاتمه ندهد، فعال باقی می‌ماند.

کد برنامه چت ساده با استفاده از پروتکل TCP/IP

مراحل پیاده‌سازی

ایجاد سرور

- ایجاد یک سوکت TCP/IP برای سرور
- تنظیم سرور برای پذیرش اتصالات ورودی از کلاینت‌ها
- دریافت پیام‌ها از کلاینت و ارسال پاسخ‌ها

ایجاد کلاینت

- ایجاد یک سوکت TCP/IP برای کلاینت
- اتصال کلاینت به سرور
- ارسال پیام‌ها به سرور و دریافت پاسخ‌ها

ارسال و دریافت پیام‌ها

- پیاده‌سازی حلقه‌های اصلی برای ارسال و دریافت پیام‌ها بین سرور و کلاینت

جزئیات فنی

زبان برنامه‌نویسی: پایتون

کتاب خانه: socket (برای ارتباط شبکه ای)

کد سمت سرور

```
import socket
# تنظیمات سرور
server_host = '127.0.0.1'  # آدرس سرور
server_port = 12345        # پورت اتصال
# ایجاد سوکت TCP
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
# باید کردن سوکت به آدرس و پورت
server_socket.bind((server_host, server_port))
# گوش دادن به اتصال (حداکثر ۱ اتصال در صف)
server_socket.listen(1)
print(f"منتظر اتصال کلاینت بر روی پورت {server_port}...")
# پذیرش اتصال از کلاینت
client_socket, client_address = server_socket.accept()
#print(f"Connected by {client_address}")
print(f"اتصال برقرار شد از: {client_address}")
try:
    # دریافت و ارسال پیام‌ها
    while True:
        # دریافت پیام از کلاینت
        data = client_socket.recv(1024).decode('utf-8')
        if not data or data.lower() == "exit":
            # اگر کلاینت ارتباط را قطع کرد
            print("ارتباط توسط کلاینت بسته شد.")
            break
        print(f"کلاینت: {data}")
```

توضیحات

۱. **وارد کردن کتابخانه** socket:

- کتابخانه socket برای ایجاد اتصالات شبکه‌ای استفاده می‌شود.

۲. **تنظیمات سرور**:

- server_host = '127.0.0.1': آدرس سرور را به لوپ‌بک (localhost)تنظیم می‌کند.
- server_port = 12345: پورت سرور را به ۱۲۳۴۵ تنظیم می‌کند.

۳. **ایجاد سوکت**:TCP

- server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM): یک سوکت TCP ایجاد می‌کند.
- server_socket.bind((server_host, server_port)): سوکت را به آدرس و پورت مشخص شده باید می‌کند.
- server_socket.listen(1): سرور را آماده پذیرش اتصال‌های ورودی می‌کند (حداکثر ۱ اتصال در صف).

```
25        # ارسال پاسخ به کلاینت
26        message = input("پیام خود را وارد کنید: ")
27        client_socket.send(message.encode('utf-8'))
28        if message.lower() == "exit":
29            print("ارتباط توسط سرور بسته شد.")
30            break
31
32    finally:
33        # بستن ارتباط
34        client_socket.close()
35        server_socket.close()
```

این کد سرور است که منتظر اتصال از سوی کلاینت می‌ماند. پس از برقراری اتصال، پیام‌ها بین سرور و کلاینت ارسال و دریافت می‌شوند.

این کد سمت کلاینت است که به سرور متصل می‌شود و پیام‌ها را به سرور ارسال می‌کند و پاسخ‌ها را دریافت می‌کند.

٤. **پذیرش اتصال از کلاینت**:

- `client_socket, client_address = server_socket.accept()`: وقتی یک کلاینت اتصال برقرار می‌کند، سرور اتصال را می‌پذیرد و سوکت کلاینت و آدرس آن را برمی‌گرداند.

٥. **دریافت و ارسال پیام‌ها**:

- یک حلقه نامحدود برای دریافت پیام از کلاینت و ارسال پاسخ به آن.
- `data = client_socket.recv(1024).decode('utf-8')`: پیام دریافتی از کلاینت را می‌خواند و رمزگشایی می‌کند.
- اگر پیام `exit` باشد یا پیام دریافتی خالی باشد، ارتباط را قطع می‌کند.
- پیام سرور را از ورودی کاربر می‌گیرد و به کلاینت ارسال می‌کند.

٦. **بستن ارتباط**:

- در نهایت، سوکت‌های کلاینت و سرور بسته می‌شوند.

کد سمت کلاینت

```
1  import socket
2  server_host = '127.0.0.1'  # IP سرور
3  server_port = 12345        # پورت اتصال
4  # ایجاد سوکت TCP
5  client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
6
7  # اتصال به سرور
8  client_socket.connect((server_host, server_port))
9  print(f"اتصال به سرور {server_host}:{server_port} برقرار شد.")
10
11 try:
12     while True:
13         # ارسال پیام به سرور
14         message = input("پیام خود را وارد کنید: ")
15         client_socket.send(message.encode('utf-8'))
16         if message.lower() == "exit":
17             print("ارتباط توسط کلاینت بسته شد.")
18             break
19
20         # دریافت پاسخ از سرور
21         data = client_socket.recv(1024).decode('utf-8')
22         if not data or data.lower() == "exit":
23             print("ارتباط توسط سرور بسته شد.")
24             break
25         print(f"سرور: {data}")
26
27 # بستن ارتباط
28 finally:
29     client_socket.close()
```

تنظیمات اولیه:

- کلاینت به آدرس IP `127.0.0.1` و پورت `12345` متصل می‌شود. آدرس `127.0.0.1` به معنای `localhost` است که ارتباط را به همان دستگاه (محلی) محدود می‌کند.

ایجاد و اتصال سوکت:

- یک سوکت TCP از نوع `socket.AF_INET` (برای IPv4) و `socket.SOCK_STREAM` (برای TCP) ایجاد می‌شود.
- با استفاده از `client_socket.connect()`، کلاینت به سرور متصل می‌شود. اگر سرور در حال اجرا باشد و به پورت و آدرس صحیح گوش دهد، اتصال موفق خواهد بود.

حلقه ارسال و دریافت پیام‌ها:

- درون یک حلقه `while`، کلاینت می‌تواند پیام‌ها را به سرور ارسال کند.
- اگر کاربر `exit` را وارد کند، کلاینت پیام را ارسال کرده و پس از آن ارتباط را خاتمه می‌دهد.
- کلاینت منتظر پاسخ از سرور می‌ماند و اگر سرور `exit` را ارسال کند یا ارتباط را ببندد، حلقه متوقف می‌شود.

بستن ارتباط:

- در بلوک `finally`، سوکت بسته می‌شود، چه ارتباط به‌طور معمول خاتمه یابد و چه به دلیل خطا متوقف شود. این عمل تضمین می‌کند که منابع به درستی آزاد می‌شوند.

اجرای برنامه

اجرای سرور

ابتدا باید برنامه سرور اجرا شود تا آماده پذیرش اتصال از کلاینت باشد.

اجرای کلاینت

سپس کد کلاینت را اجرا کنید تا به سرور متصل شود و پیام‌ها رد و بدل شوند.

خاتمه ارتباط: با تایپ کردن exit در ورودی سرور یا کلاینت، ارتباط به صورت ایمن خاتمه می‌یابد.