یک **سرور TCP ساده** را پیاده‌سازی می‌کند که ارتباطات شبکه را از طریق پروتکل TCP مدیریت می‌کند. در این کد، فرآیند به شرح زیر است:

1. **ساخت سرور TCP:**
   * ابتدا با استفاده از TcpListener::bind("127.0.0.1:7878") یک سرور بر روی IP 127.0.0.1 و پورت 7878 راه‌اندازی می‌شود که به اتصالات ورودی گوش می‌دهد.
2. **پذیرش اتصالات:**
   * سرور به صورت مداوم منتظر اتصالات جدید از کلاینت‌ها می‌ماند. هر زمان که یک اتصال جدید برقرار می‌شود، با استفاده از listener.incoming()، اتصالی را دریافت می‌کند.
3. **پردازش درخواست‌ها:**
   * زمانی که یک اتصال جدید از طرف یک کلاینت برقرار می‌شود، یک نخ (thread) جدید به نام handle\_client ایجاد می‌شود.
   * این نخ مسئول پردازش داده‌های دریافتی از کلاینت است. داده‌ها از طریق stream.read(&mut buffer) خوانده شده و سپس با استفاده از stream.write\_all(&buffer[0..n]) به همان شکل به کلاینت باز می‌گردد (عملکرد Echo).
4. **پایان اتصال:**
   * وقتی داده‌ای از کلاینت دریافت نشود (Ok(0) که نشان‌دهنده بستن اتصال است)، ارتباط قطع می‌شود و نخ خاتمه می‌یابد.
5. **اجرای برنامه:**
   * سرور پس از شروع به کار، به صورت مداوم آماده پذیرش اتصالات است و تا زمانی که برنامه در حال اجرا باشد، این فرآیند ادامه دارد.

این برنامه یک پیاده‌سازی ساده از **برنامه‌نویسی شبکه با Rust** است که مفهوم **اتصالات TCP** و **چندنخ‌کاری (Multithreading)** را به خوبی نشان می‌دهد.