Verteilte Systeme

...für C++ Programmierer TCP/IP Programmierung 3 - Server (synchron)

by

Dr. Günter Kolousek

23. Januar 2021

Passive Sockets (Server)

1. Anlegen

```
tcp::endpoint ep{tcp::v4(), 9999};
tcp::acceptor acceptor{ctx};
```

2. Öffnen

```
// also with error_code (as usually)
acceptor.open(ep.protocol());
// 1 & 2:
//tcp::acceptor acceptor{ctx,ep.protocol()};
// → third param: reuse_addr=true !!
// but this can throw an exception!
```

3. Setzen von Optionen (optional)

```
// allows to bind even if in TIME-WAIT
acceptor.set_option(
  tcp::acceptor::reuse_address{true});
```

Passive Sockets (Server) - 2

4. Binden

```
acceptor.bind(ep);
// 1, 2 & 4
// tcp::acceptor acceptor{ctx, ep};
```

5. Listen

```
// unnecessary because it's the default ↓
acceptor.listen(tcp::socket::max_connections)
```

6. Verbindungsanfrage akzeptieren

```
tcp::socket sock{ctx};  // create active sock
// use this socket for next connection
acceptor.accept(sock);  // blocking!
```

7. Schließen (wenn keine weitere Verbindung)
   - `acceptor.close()` … schließen

Synchroner Echo-Server

```cpp
#include <iostream>  // sync_echo_server.cpp
#include <asio.hpp>
using namespace std; using namespace asio::ip;
int main() { asio::io_context ctx;
    tcp::endpoint ep{tcp::v4(), 9999};
    tcp::acceptor acceptor{ctx, ep};
    acceptor.listen();
    tcp::socket sock{ctx};
    acceptor.accept(sock);
    // from now no further accept possible
    acceptor.close();
```

Synchroner Echo-Server - 2

```
asio::streambuf buf;
asio::read_until(sock, buf, '\n');
string reply;
istream is{&buf};
getline(is, reply);
asio::write(sock, asio::buffer(reply,
    reply.size()));
cout << "sent: " << reply << endl;
cout << "local port: "  // → 9999
  << sock.local_endpoint().port() << endl;
cout << "remote port: "
  << sock.remote_endpoint().port() << endl;
sock.close();  }
```

Synchroner MT Echo-Server

```
#include <iostream>  // mt_sync_echo_server.cpp
#include <thread>
#include <asio.hpp>
using namespace std; using namespace asio::ip;
void serve_client(tcp::socket&& sock) {
    asio::streambuf buf;
    asio::read_until(sock, buf, '\n');
    string reply;
    istream is{&buf};
    getline(is, reply);
    asio::write(sock, asio::buffer(reply,
        reply.size()));
    cout << ”sent: ” << reply << endl;
    sock.close();
}
```

Synchroner MT Echo-Server - 2

```
int main() { asio::io_context ctx;
    tcp::endpoint ep{tcp::v4(), 9999};
    tcp::acceptor acceptor{ctx, ep};
    acceptor.listen();
    while (true) {
        tcp::socket sock{ctx};
        acceptor.accept(sock);
        thread thd{serve_client, move(sock)};
        thd.detach();
    }
  }
```