

3-2 加餐 - connect 的断线重连

客户端会面临服务器崩溃的情况，我们可以试着写一个客户端重连的代码，模拟并理解一些客户端行为，比如游戏客户端等

完整代码链接：<https://gitee.com/whb-helloworld/linux-plus-meal/tree/master/disconnect-and-reconnect>

TcpClient.cc

- 采用状态机，实现一个简单的 tcp client 可以实现重连效果

```
C++
#include <iostream>
#include <string>
#include <cstring>
#include <cstdlib>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

using namespace std;

void Usage(const std::string &process)
{
    std::cout << "Usage: " << process << " server_ip server_port"
<< std::endl;
}

enum class Status // C++11 强类型枚举
{
    NEW,                // 新建状态，就是单纯的连接
    CONNECTING,         // 正在连接，仅仅方便查询 conn 状态
    CONNECTED,          // 连接或者重连成功
    DISCONNECTED,       // 重连失败
    CLOSED              // 连接失败，经历重连，无法连接
};

class ClientConnection
```

```

{
public:
    ClientConnection(uint16_t serverport, const std::string
&serverip)
        : _sockfd(-1),
          _serverport(serverport),
          _serverip(serverip),
          _retry_interval(1),
          _max_retries(5),
          _status(Status::NEW)
    {
    }
    void Connect()
    {
        // 1. 创建 socket
        _sockfd = socket(AF_INET, SOCK_STREAM, 0);
        if (_sockfd < 0)
        {
            cerr << "socket error" << endl;
            exit(1);
        }

        // 2. 要不要 bind? 必须要有 Ip 和 Port, 需要 bind, 但是不需要用
        // 户显示的 bind, client 系统随机端口
        // 发起连接的时候, client 会被 OS 自动进行本地绑定
        // 2. connect
        struct sockaddr_in server;
        memset(&server, 0, sizeof(server));
        server.sin_family = AF_INET;
        server.sin_port = htons(_serverport);
        // p:process(进程), n(网络) -- 不太准确, 但是好记忆
        inet_pton(AF_INET, _serverip.c_str(), &server.sin_addr);
        // 1. 字符串 ip->4 字节 IP 2. 网络序列

        int n = connect(_sockfd, (struct sockaddr *)&server,
sizeof(server)); // 自动进行 bind 哦!
        if (n < 0)
        {
            Disconnect(); // 恢复_sockfd 的默认
            // 值, 是连接没有成功, 不代表 sockfd 创建没有成功
            _status = Status::DISCONNECTED; // 没有连接成功
            return;
        }
    }
}

```

```

        _status = Status::CONNECTED; // 连接成功
    }
    int SocketFd()
    {
        return _sockfd;
    }
    void Reconnect()
    {
        _status = Status::CONNECTING; // 正在重连
        int count = 0;
        while (count < _max_retries)
        {
            Connect(); // 重连
            if (_status == Status::CONNECTED)
            {
                return;
            }
            sleep(_retry_interval);
            count++;
            std::cout << "重连次数: " << count << ", 最大上限: " <<
_max_retries << std::endl;
        }
        _status = Status::CLOSED; // 重连失败, 可以关闭了
    }
    void Disconnect()
    {
        if (_sockfd != -1)
        {
            close(_sockfd);
            _status = Status::CLOSED;
            _sockfd = -1;
        }
    }
    Status GetStatus()
    {
        return _status;
    }
    void Process()
    {
        // 简单的 IO 即可
        while (true)
        {
            string inbuffer;
            cout << "Please Enter# ";

```

```

        getline(cin, inbuffer);
        if(inbuffer.empty()) continue;

        ssize_t n = write(_sockfd, inbuffer.c_str(),
inbuffer.size());
        if (n > 0)
        {
            char buffer[1024];
            ssize_t m = read(_sockfd, buffer, sizeof(buffer) -
1);

            if (m > 0)
            {
                buffer[m] = 0;
                cout << "echo messsge -> " << buffer << endl;
            }
            else if (m == 0) // 这里证明 server 端掉线了
            {
                _status = Status::DISCONNECTED;
                break;
            }
            else
            {
                std::cout << "read m : " << m << "errno: " <<
errno << "errno string: " << strerror(errno) << std::endl;
                _status = Status::CLOSED;
                break;
            }
        }
        else
        {
            std::cout << "write n : " << n << "errno: " <<
errno << "errno string: " << strerror(errno) << std::endl;
            _status = Status::CLOSED;
            break;
        }
    }

    ~ClientConnection()
    {
        Disconnect();
    }

private:
    int _sockfd;

```

```

uint16_t _serverport; // server port 端口号
std::string _serverip; // server ip 地址
int _retry_interval; // 重试时间间隔
int _max_retries; // 重试次数
Status _status; // 连接状态
};

class TcpClient
{
public:
    TcpClient(uint16_t serverport, const std::string &serverip) :
        _conn(serverport, serverip)
    {
    }
    void Execute()
    {
        while (true)
        {
            switch (_conn.GetStatus())
            {
            case Status::NEW:
                _conn.Connect();
                break;
            case Status::CONNECTED:
                std::cout << "连接成功, 开始进行通信." << std::endl;
                _conn.Process();
                break;
            case Status::DISCONNECTED:
                std::cout << "连接失败或者对方掉线, 开始重连." <<
std::endl;
                _conn.Reconnect();
                break;
            case Status::CLOSED:
                _conn.Disconnect();
                std::cout << "重连失败, 退出." << std::endl;
                return; // 退出
            default:
                break;
            }
        }
    }
    ~TcpClient()
    {

```

```


    }

private:
    ClientConnection _conn; // 简单组合起来即可
};
// class Tcp

// ./tcp_client serverip serverport
int main(int argc, char *argv[])
{
    if (argc != 3)
    {
        Usage(argv[0]);
        return 1;
    }
    std::string serverip = argv[1];
    uint16_t serverport = stoi(argv[2]);
    TcpClient client(serverport, serverip);
    client.Execute();
    return 0;
}

```

重连成功



```

whb@bite01:~/linux-plus-meal/disconnect-and-reconnect$ ./tcpclient 127.0.0.1 8888
连接失败或者对方掉线, 开始重连
重连次数: 1, 最大上限: 5
重连次数: 2, 最大上限: 5
重连次数: 3, 最大上限: 5
连接成功, 开始进行通信。
Please Enter# hello
echo message -> hello[tcp server echo message]
Please Enter# world
echo message -> world[tcp server echo message]
Please Enter# a
连接失败或者对方掉线, 开始重连。
重连次数: 1, 最大上限: 5
重连次数: 2, 最大上限: 5
连接成功, 开始进行通信。
Please Enter#

```

```

whb@bite01:~/linux-plus-meal/windows-udp-tcp$ ./tcp_server 8888
get a new connection, info is : 127.0.0.1:37782
127.0.0.1:37782# hello
127.0.0.1:37782# world
^C
关闭服务器
whb@bite01:~/linux-plus-meal/windows-udp-tcp$ ./tcp_server 8888
get a new connection, info is : 127.0.0.1:37796

```

重连失败

连接失败或者对方掉线, 开始重连。

重连次数: 1, 最大上限: 5

重连次数: 2, 最大上限: 5

重连次数: 3, 最大上限: 5

重连次数: 4, 最大上限: 5

重连次数: 5, 最大上限: 5

重连失败, 退出。