12-1 加餐 - poll 代码改写

附录：

- 结合 select 代码，将 select server 更改成为 pollserver，不是一件困难的事情
- 测试代码链接：https://gitee.com/whb-helloworld/linux-plus-meal/tree/master/select-demo

```C++
#pragma once

#include <iostream>
#include <string>
#include <poll.h>
#include <memory>
#include "Log.hpp"
#include "Socket.hpp"

using namespace Net_Work;

const static int gdefaultport = 8888;
const static int gbacklog = 8;
const int gnum = 1024;

class PollServer
{
private:
    void HandlerEvent()
    {
        for (int i = 0; i < _num; i++)
        {
            if (_rfds[i].fd == -1)
                continue;
            // 合法的 sockfd
            // 读事件分两类，一类是新连接到来。 一类是新数据到来
            int fd = _rfds[i].fd;
            short revents = _rfds[i].revents;

            if (revents & POLLIN)
            {
```

```
                // 新连接到来了
                if (fd == _listensock->GetSockFd())
                {
                    lg.LogMessage(Info, "get a new link\n");
                    // 获取连接
                    std::string clientip;
                    uint16_t clientport;
                    // 不会阻塞！！，因为 select 已经检测到了
listensock 已经就绪了
                    int sock = _listensock-
>AcceptConnection(&clientip, &clientport);
                    if (sock == -1)
                    {
                        lg.LogMessage(Error, "accept error\n");
                        continue;
                    }
                    lg.LogMessage(Info, "get a client, client info
is# %s:%d, fd: %d\n", clientip.c_str(), clientport, sock);

                    // 这里已经获取连接成功了，接下来怎么办？？？
                    // read? write? 绝对不能！！！read 底层数据是否就
绪时不确定的！谁清楚 fd 上面是否有读事件呢？poll!
                    // 新链接 fd 到来的时候，要把新的 fd，想办法交给
poll 托管 -- 只需要添加到数组_rfds 中即可
                    int pos = 0;
                    for (; pos < _num; pos++)
                    {
                        if (_rfds[pos].fd == -1)
                        {
                            _rfds[pos].fd = sock;
                            _rfds[pos].events = POLLIN;
                            lg.LogMessage(Info, "get a new link,
fd is : %d\n", sock);
                            break;
                        }
                    }
                    if (pos == _num)
                    {
                        // 1. 扩容
                        // 2. 关闭
                        close(sock);
                        lg.LogMessage(Warning, "server is
full...!\n");
```

```
                    }
                }
                else
                {
                    // 普通的读事件就绪
                    // 读数据是有问题的
                    // 这一次读取不会被卡住吗？
                    char buffer[1024];
                    ssize_t n = recv(fd, buffer, sizeof(buffer-1),
0); // 这里读取会阻塞吗？不会！
                    if (n > 0)
                    {
                        buffer[n] = 0;
                        lg.LogMessage(Info, "client say# %s\n",
buffer);
                        std::string message = "你好呀，少年，";
                        message += buffer;
                        send(fd, message.c_str(), message.size(),
0);
                    }
                    else
                    {
                        lg.LogMessage(Warning, "client quit, maybe
close or error, close fd : %d\n", fd);
                        close(fd);
                        // 取消 poll 的关心
                        _rfds[i].fd = -1;
                        _rfds[i].events = 0;
                        _rfds[i].revents = 0;
                    }
                }
            }
        }
    }

public:
    PollServer(int port = gdefaultport) : _port(port),
_listensock(new TcpSocket()), _isrunning(false), _num(gnum)
    {
    }
    void InitServer()
    {
        _listensock->BuildListenSocketMethod(_port, gbacklog);
        _rfds = new struct pollfd[_num];
```

```
        for (int i = 0; i < _num; i++)
        {
            _rfds[i].fd = -1;
            _rfds[i].events = 0;
            _rfds[i].revents = 0;
        }
        // 最开始的时候，只有一个文件描述符， Listensock
        _rfds[0].fd = _listensock->GetSockFd();
        _rfds[0].events |= POLLIN;
    }
    void Loop()
    {
        _isrunning = true;
        while (_isrunning)
        {
            // 定义时间
            int timeout = -1;
            // rfds 本质是一个输入输出型参数，rfds 是在 select 调用返回
的时候，不断被修改，所以，每次都要重置
            PrintDebug();
            int n = poll(_rfds, _num, timeout);
            switch (n)
            {
            case 0:
                lg.LogMessage(Info, "poll timeout...\n");
                break;
            case -1:
                lg.LogMessage(Error, "poll error!!!\n");
                break;
            default:
                // 正常的就绪的 fd
                lg.LogMessage(Info, "select success, begin event
handler\n");
                HandlerEvent(); // _rfds_array: 3,4,5,6,7,8,9,10 -
> rfds: 4,5,6
                break;
            }
        }
        _isrunning = false;
    }
    void Stop()
    {
        _isrunning = false;
    }
```

```
    void PrintDebug()
    {
        std::cout << "current poll fd list is : ";
        for (int i = 0; i < _num; i++)
        {
            if (_rfds[i].fd == -1)
                continue;
            else
                std::cout << _rfds[i].fd << " ";
        }
        std::cout << std::endl;
    }
    ~PollServer()
    {
        delete[] _rfds;
    }

private:
    std::unique_ptr<Socket> _listensock;
    int _port;
    int _isrunning;
    struct pollfd *_rfds;
    int _num;
};
```