Bug修改记录

1.19477：LTcpSocket的listens 状态错误

- 原因：`Tcp`服务端在监听之前需要先判断是否绑定了端口号和地址
- 解决：在`LAbstractSocket`当中添加了`isBind`标签记录是否绑定，并且在抽象类的源文件当中做对`isBind`标签状态的维护

  头文件：

```
/**
 * @brief 存储是否绑定了IP和端口
 *
 */
bool isBind = false;
```

  源文件：

```
bool LAbstractSocket::binds(const LHostAddress &address, in_port_t port) {
    setLocalAddress(address);
    setLocalPort(port);

    int ret = -1;
    if (pData.protocol == IPv4Protocol) {
        struct sockaddr_in bindadd;
        bindadd.sin_family = AF_INET;
        bindadd.sin_port = htons(local.port);
        bindadd.sin_addr = local.add.getIPv4();
        socklen_t len = sizeof(struct sockaddr_in);

        ret = bind(socketfd, (struct sockaddr *)&bindadd, sizeof(bindadd));

    } else if (pData.protocol == IPv6Protocol) {
        struct sockaddr_in6 bindadd;
        bindadd.sin6_family = AF_INET6;
        bindadd.sin6_port = htons(local.port);
        bindadd.sin6_addr = local.add.getIPv6();
        socklen_t len = sizeof(struct sockaddr_in6);

        ret = bind(socketfd, (struct sockaddr *)&bindadd, sizeof(bindadd));
    } else {
        std::cerr << "Binds error : please set NetworkProtocol" << std::endl;
        return false;
    }

    if (-1 == ret) {
        std::cerr << "Binds error : creat failed" << std::endl;
        return false;
    }

    // 修改isBind标签
    isBind = true;

    return true;
}
```

```cpp
bool LAbstractSocket::closes() {
    if (socketfd > 0) {
        close(socketfd);
        socketfd = -1;
        isBind = false;

        return true;
    } else {
        std::cerr << "socket未连接" << std::endl;
        return false;
    }
}
```

同时`LTcpSocket`的`listens()`函数当中也做对应修改：

```cpp
bool LTcpSocket::listens(int backlog = 5) {
    if (!isBind) {
        std::cerr << "未调用binds函数绑定IP和端口号!" << std::endl;
        return false;
    }

    int ret = listen(localfd(), backlog);
    if (ret < 0 or backlog < 0) {
        std::cerr << "listens error" << std::endl;
        return false;
    }
    return true;
}
```

- 测试：在`LTcpClientDemo`的`test_5()`中

```
lark5@DavidingPlus:~/Lark5/larksdk/build$ ./snippet/LTCPClientDemo/LTCPClientDemo
未调用binds函数绑定IP和端口号!
0
1
1
lark5@DavidingPlus:~/Lark5/larksdk/build$
```

2.关于几个exit()函数

- 解决：根据要求改为抛出异常
  - `labstractsocket.cpp`：

```cpp
// 读取缓冲区可读取字节数
if (-1 == ioctl(socketfd, FIONREAD, &bytesAvailable))
    throw LException("读取缓冲区待读取字节数错误!");
```

  - `lhttpcontrol.cpp`：

```cpp
int len = recv(connectFd, read_buf, BUFSIZ - 1, 0);
if (-1 == len)
    throw LException("接收响应报文失败(recv error)");
```

```
void LHttpControl::dealSigAlrm(int num) {
    throw LException("connect连接超时!请检查您输入的地址!");
}
```

- `ltcpsocket.cpp`:

```
if (-1 == ret)
    throw LException("接收数据失败(recv error)");
```

- `ludpsocket.cpp`:

```
len);
if (-1 == ret)
    throw LException("接收数据失败(recvfrom error)");
```

- 再来看`bug`，我都做了捕捉异常的处理，看是否能打印出正确的信息
  - **19448：LTcpSocket的客户端断开连接后使用receives 程序报错并直接退出**

    测试程序在`LTCPCLientDemo`的`test_6()`中

```
lark5@DavidingPlus:~/Lark5/larksdk/build$ ./snippet/LTCPClientDemo/LTCPClientDemo
接收数据失败(recv error)
lark5@DavidingPlus:~/Lark5/larksdk/build$
```

  - **19442，19443：都是connect的exit()，放在一起了**

    测试程序在`HttpControlTest`的`testDelete3()`中

```
lark5@DavidingPlus:~/Lark5/larksdk/build$ ./snippet/HttpControlTest/HttpControlTest
connect连接超时!请检查您输入的地址!
1
lark5@DavidingPlus:~/Lark5/larksdk/build$
```

3.`buffersize`这个问题

- 解决：将`TCP`里面的已经把循环处理给删除了

  现在的情况就是，发送方发送的数据很长，但是由于接收方设置的存储区大小很小，所以没办法一次接收完毕，需要用户手动循环处理，多余的数据烂在接收缓冲区了，这也是符合常识的

```cpp
int LTcpSocket::receives() {
    if (bufferSize() <= 0) {
        std::string message = std::string("缓冲区大小 ") + std::to_string(bufferSize()) +
        std::string(" 不合理,请检查并且重试!");

        throw LException(LString(message));
    }

    // 测试要求不做循环读取，意思是让bytebuffer最大的容量只能是buffersize，多出的数据下次手动receives的
    时候再读取，或者不读就烂在接收缓冲区里 ...
    char buf[pData.buffersize + 1] = {0};  // 多开一个'\0'

    int ret = recv(localfd(), buf, pData.buffersize, 0);
    if (-1 == ret)
        throw LException("接收数据失败(recv error)");

    // 接收字符串
    pData.bytebuffer = std::string(buf);

    return ret;
}
```

- 测试：在`LTCPDemo`的`test4()`中

  读两次，得到的数据是符合预期的

```
lark5@DavidingPlus:~/Lark5/larksdk/build$ ./snippet/LTCPServerDemo/LTCPServerDemo
recv: he
recv: ll
lark5@DavidingPlus:~/Lark5/larksdk/build$
```

```
lark5@DavidingPlus:~/Lark5/larksdk/build$ ./snippet/LTCPClientDemo/LTCPClientDemo
sends: hello,i am client,got it...1231
lark5@DavidingPlus:~/Lark5/larksdk/build$
```