Bug修改记录 11.9

1.19258：LHostAddress类未设置ip地址，直接调用getAddress导致后续无输出

- 解决和原因：`getAddress`的返回值很抽象，返回一个`char *`，还是`new`出来的，我不知道学长怎么想的，至少我不会返回字符串返回`char *`指针，我改成`LString`了

```
/**
 * @brief 获取用字符串表示的地址（网络字节序）
 */
LString LHostAddress::getAddress() {
    // 获取不到地址就是type类型为Unknown类型
    if (Unknown == pAddr.type)
        return LString();

    if (pAddr.type == IPv6addr) {
        struct in6_addr addr;
        char* ipv6_str = new char[INET6_ADDRSTRLEN];
        inet_ntop(AF_INET6, &(pAddr.ipv6), ipv6_str, INET6_ADDRSTRLEN);

        return LString(ipv6_str);
    } else if (pAddr.type == IPv4addr) {
        char* ipv4_str = new char[INET_ADDRSTRLEN];
        inet_ntop(AF_INET, &(pAddr.ipv4), ipv4_str, INET_ADDRSTRLEN);

        return LString(ipv4_str);
    }
    return LString();
}
```

- 测试：在`LHostAddressTest`的`test5()`中

```
lark5@DavidingPlus:~/Lark5/larksdk/build$ ./snippet/LHostAddressTest/LHostAddressTest
未设置前的地址为:
设置的IP地址为: 127.0.0.1
清空后的IP地址为:
lark5@DavidingPlus:~/Lark5/larksdk/build$
```

2.19250： LAbstractSocket的setLocalAddress函数传入nullptr 崩溃

- 原因：`setLocalAddress`函数没有对传入空指针做判断，导致内存非法访问，就段错误了
- 解决：分别做上判断，如果空指针，抛出异常

```
 * @param add 想要绑定的地址
 */
void LAbstractSocket::setLocalAddress(LHostAddress *add) {
    if (nullptr == add)
        throw LException("传入空指针,setLocalAddress错误!");

    local.add = *add;
}

/**
 * @brief 设置本地主机端口号，为主机字节序
 * @param port uint16_t类型，想要绑定的端口号
 */
void LAbstractSocket::setLocalPort(uint16_t port) {
    local.port = port;
}

/**
 * @brief 设置连接方主机地址
 * @param add 想要绑定的地址
 */
void LAbstractSocket::setPeerAddress(LHostAddress *add) {
    if (nullptr == add)
        throw LException("传入空指针,setPeerAddress错误!");

    peer.add = *add;
}
```

- 测试：`LTCPCLientDemo`的`test5()`，我捕捉异常后后续程序正常

```
lark5@DavidingPlus:~/Lark5/larksdk/build$ ./snippet/LTCPClientDemo/LTCPClientDemo
传入空指针,setLocalAddress错误!
127.0.0.1
lark5@DavidingPlus:~/Lark5/larksdk/build$
```

3.19275和19276：关于连接不存在的地址的问题

- 分析：分为两种情况，在`connect`这里会有一点区别
  - 第一种是正确的`IP`，这种情况下如果端口也正确，那么`connect`会成功；如果端口不正确，那么会立即返回-1，表示连接失败，这个原先的代码已经做了正确的抛出异常的处理；如果路径不正确，那么服务端这边会处理，会返回404，这个就是正确的接受
  - 第二种是IP就不正确，这样`connect`根本找不到服务端，由于`connect`默认是阻塞的，他会一直阻塞尝试去连接，所以看起来就会卡死，由于`connect`默认的超时时间是75秒到几分钟(我不知道具体的，我查的资料)，太长了，所以我们这里需要改造一下，让他更快的响应，我采用的是通过定时器，然后捕捉`SIGALRM`信号

    具体如下：

```cpp
// 捕捉信号
struct sigaction act;
act.sa_flags = 0;
act.sa_handler = deal_sigalrm;

sigaction(SIGALRM, &act, nullptr);

// 注册定时器
alarm(overtime);

if (connect(sockfd, (struct sockaddr *)&servaddr, sizeof(servaddr)) < 0) {
    reply->setError(reply, LHttpReply::ConnectionRefusedError);
    throw LException("连接到服务器失败(connect error)");
}
// 销毁定时器
alarm(0);
```

定义的私有成员：

```cpp
/**
 * @brief 对捕捉到的SIGALRM信号进行处理
 * @param num ，信号的编号或者宏
 */
static void deal_sigalrm(int num);

/**
 * @brief 定义connect连接我们设定的超时时间
 *
 */
int overtime = 15;
```

- 测试：在`HttpControlTest`的`testDelete2`和`testDelete3`中，2是`IP`正确但是路径不正确，3是`IP`不对，也就是会超时

  2结果如下：

```
lark5@DavidingPlus:~/Lark5/larksdk/build$ ./snippet/HttpControlTest/HttpControlTest
StatusCode:404 Not Found
-------------------------
SendMessage:**************
DELETE /10086 HTTP/1.1
Host: 192.168.1.211
User-Agent: Mozilla/5.0
Accept: */*
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6
Connection: keep-alive
```

  3结果如下：

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