欢迎大家来到第五阶段课程

《分布式流媒体》实训项目

TNV DAY08

复习课

预习
内容

客户机（6）

客户机（6）

连接类(conn_c)的二级方法

- 打开连接：open
  - 创建连接对象
  - 连接目的主机
- 关闭连接：close
  - 销毁连接对象
- 构造请求：makerequ
  - 在请求缓冲区中填入命令、状态、应用ID、用户ID和文件ID

连接类(conn_c)的二级方法

- 接收包体：recvbody
  - 接收包头
    - ➢ 既非本地错误亦非套接字通信错误且包体非空
      - 分配包体
      - 接收包体
  - 返回处理结果
- 接收包头：recvhead
  - 接收包头
  - 解析包头
    - ➢ 检查并输出包体长度
    - ➢ 检查状态
  - 返回处理结果

附录：程序清单

TNV/src/05_client/02_conn.cpp

```
//打开连接
bool conn_c::open(void) {
        if (m_conn)
                return true;

        //创建连接对象
        m_conn = new acl::socket_stream;

        //连接目的主机
        if (!m_conn->open(m_destaddr, m_ctimeout, m_rtimeout)) {
                logger_error("open %s fail: %s",
                        m_destaddr, acl_last_serror());
                delete m_conn;
                m_conn = NULL;
                m_errnumb = -1;
```

TNV/src/05_client/02_conn.cpp

```
                m_errdesc.format("open %s fail: %s",
                        m_destaddr, acl_last_serror());
                return false;
        }

        return true;
}

//关闭连接
void conn_c::close(void) {
        if (m_conn) {
                delete m_conn;
                m_conn = NULL;
        }
}
```

TNV/src/05_client/02_conn.cpp

```
// 构造请求
int conn_c::makerequ(char command, char const* appid,
        char const* userid, char const* fileid, char* requ) {
        // |包体长度|命令|状态|应用ID|用户ID|文件ID|
        // |    8    |  1 |  1 |  16   |  256  |  128  |
        requ[BODYLEN_SIZE] = command; // 命令
        requ[BODYLEN_SIZE+COMMAND_SIZE] = 0; // 状态

        // 应用ID
        if (strlen(appid) >= APPID_SIZE) {
                logger_error("appid too big, %lu >= %d",
                        strlen(appid), APPID_SIZE);
                m_errnumb = -1;
                m_errdesc.format("appid too big, %lu >= %d",
                        strlen(appid), APPID_SIZE);
```

TNV/src/05_client/02_conn.cpp

知识讲解

```
        return ERROR;
}
strcpy(requ + HEADLEN, appid);

//用户ID
if (strlen(userid) >= USERID_SIZE) {
        logger_error("userid too big, %lu >= %d",
                strlen(userid), USERID_SIZE);
        m_errnumb = -1;
        m_errdesc.format("userid too big, %lu >= %d",
                strlen(userid), USERID_SIZE);
        return ERROR;
}
strcpy(requ + HEADLEN + APPID_SIZE, userid);
```

知识讲解

TNV/src/05_client/02_conn.cpp

```
        // 文件ID
        if (strlen(fileid) >= FILEID_SIZE) {
                logger_error("fileid too big, %lu >= %d",
                        strlen(fileid), FILEID_SIZE);
                m_errnumb = -1;
                m_errdesc.format("fileid too big, %lu >= %d",
                        strlen(fileid), FILEID_SIZE);
                return ERROR;
        }
        strcpy(requ + HEADLEN + APPID_SIZE + USERID_SIZE, fileid);

        return OK;
}

// 接收包体
```

知识讲解

TNV/src/05_client/02_conn.cpp

```
int conn_c::recvbody(char** body, long long* bodylen) {
        //接收包头
        int result = recvhead(bodylen);

        //既非本地错误亦非套接字通信错误且包体非空
        if (result != ERROR && result != SOCKET_ERROR && *bodylen) {
                //分配包体
                if (!(*body = (char*)malloc(*bodylen))) {
                        logger_error("call malloc fail: %s, bodylen: %lld",
                                strerror(errno), *bodylen);
                        m_errnumb = -1;
                        m_errdesc.format("call malloc fail: %s, bodylen: %lld",
                                strerror(errno), *bodylen);
                        return ERROR;
                }
```

TNV/src/05_client/02_conn.cpp

知识讲解

```
                //接收包体
                if (m_conn->read(*body, *bodylen) < 0) {
                        logger_error("read fail: %s, from: %s",
                                acl::last_serror(), m_conn->get_peer());
                        m_errnumb = -1;
                        m_errdesc.format("read fail: %s, from: %s",
                                acl::last_serror(), m_conn->get_peer());
                        free(*body);
                        *body = NULL;
                        close();
                        return SOCKET_ERROR;
                }
        }

        return result;
}
```

TNV/src/05_client/02_conn.cpp

```
//接收包头
int conn_c::recvhead(long long* bodylen) {
        if (!open())
                return SOCKET_ERROR;

        char head[HEADLEN]; //包头缓冲区

        //接收包头
        if (m_conn->read(head, HEADLEN) < 0) {
                logger_error("read fail: %s, from: %s",
                        acl::last_serror(), m_conn->get_peer());
                m_errnumb = -1;
                m_errdesc.format("read fail: %s, from: %s",
                        acl::last_serror(), m_conn->get_peer());
                close();
```

TNV/src/05_client/02_conn.cpp

```
                return SOCKET_ERROR;
        }

        // |包体长度|命令|状态|
        // |    8   |  1 |  1 |
        //解析包头
        if ((*bodylen = ntoll(head)) < 0) { //包体长度
                logger_error("invalid body length: %lld < 0, from: %s",
                        *bodylen, m_conn->get_peer());
                m_errnumb = -1;
                m_errdesc.format("invalid body length: %lld < 0, from: %s",
                        *bodylen, m_conn->get_peer());
                return ERROR;
```

TNV/src/05_client/02_conn.cpp

知识讲解

```
        }
        int command = head[BODYLEN_SIZE]; // 命令
        int status = head[BODYLEN_SIZE+COMMAND_SIZE]; // 状态
        if (status) {
                logger_error("response status %d != 0, from: %s",
                        status, m_conn->get_peer());
                return STATUS_ERROR;
        }
        logger("bodylen: %lld, command: %d, status: %d",
                *bodylen, command, status);

        return OK;
}
```

下节课见