Linux 下基于 socket 多线程并发通信系统设计与开发*

王珂珑　　林宁**

南宁学院信息工程学院，南宁 530200

摘 要　本系统在 Linux 桌面系统 Ubuntu 下采用 C/S 模式开发的即使通信系统程序，系统使用 C++编程语言为主，通信技术使用的是 Linux 系统下的 socket 的 C 语言接口,通信协议采用的是 TCP/IP 为主的网络通信协议[1]，服务器采用的是多线程技术来完成多个连接请求任务的处理，其中还是用了 Linux 内核提供的 epoll 即 IO 多路复用技术来提高系统服务器的并发数量，系统客户端主要采用的 QT 框架作为客户端图形界面开发框架，利用这些技术最终实现系统的用户的登录注册、好友的添加搜索删除、群聊的创建、删除群聊、加入群聊、在线群聊、在线私聊等基本功能。

关键词　即时通信；C/C++；Linux；多线程并发；Qt

Linux is Based on Socket Multi-Threaded Concurrent Communication Letter Design and Development

Wang kelong　Lin Ning

School of Information Engineering
Nanning University
Nanning 530200,China;

Abstract—This system is developed in the C/S mode under the Linux desktop system Ubuntu, even though the communication system program mainly uses the C++programming language. The communication technology uses the C language interface of the socket under the Linux system, and the communication protocol uses the network communication protocol mainly based on TCP/IP. The server uses the multithreading technology to complete the processing of multiple connection request tasks, The EPOLL (IO Multiplexing) technology provided by the Linux kernel is also used to increase the concurrent number of system servers. The QT framework is mainly used as the client graphical interface development framework for the system client, and these technologies are ultimately used to achieve basic functions such as login and registration of system users, addition, search, deletion, creation, deletion, joining, online group chat, and online private chat.

Keyword—Instant messaging, C/C++, Linux, Multi thread concurrency, Qt

1 引　言

现如今随着 Linux 桌面系统用户逐渐的增多，在这样的发展洪流下在 Linux 系统中对即时通信的需求日益增多,而在 Linux 中没有很好的即时通信软件,所以研究和开发一个易用性和高性能的通信系统软件是非常有必要的，即时通信软件可以大大的提高了用户的办公效率，可以通过即时通信系统软件进行沟通和探讨相关的技术的问题，从而提高在 Linux 系统环境下的办公效率和信息沟通效率。

*基金资助：本文得到广西高校中青年教师科研基础能力提升项目（2021KY1805）资助。
**通讯作者：林宁，副教授，bgy_2009@163.com

2 相关技术与工具介绍

2.1 Linux socket 网络编程接口介绍

Socket 是通过给出其他计算机的端口，再通过该端口与其他的 socket 接口的程序进行通信,应用程序再通过 socket 接口在网络上进行数据的传输、数据接收，系统采用的是 Linux 提供的 Socket 函数，在通过 C++将其过程封装成不同的类作为模块,方便后期管理和模块化编程,结合其他底层 Linux 函数以实现 Linux 操作系统平台的通信程序工具。

2.2 Mysql 数据库连接池

系统采用的是常见的数据库 MySQL，MySQL 提供了 C 语言的 mysql_real_connect()函数 api 来对数据

库进行连接，通过 C++利用面向对象的思想对数据库的连接进行一个完备的封装，将数据库的连接操作封装成了一个数据库连接池，使用数据库连接池的原因主要是为了解决数据库操作时带来的资源创建和释放所带来的性能损耗问题，通过使用数据库连接池技术可以很好的对数据库连接进行一个管理、分配和释放，从而实现系统的数据库连接的重用，数据库连接池主要的实现方式，通过将数据库连接封装成一个类，利用 C++提供的 STL 容器将其存储，当要使用到数据库操作时就将其从中取出，当数据库操作完成后再将其放回到容器中，实现重用的效果，从而减少对连接的创建和销毁等操作。

2．3 epoll 的反应堆模型

Reactor 模型是处理并发 IO 最常见的一种模式，处理并发的主要思路是，将所有要处理的 IO 事件注册到一个主线程 IO 的多路复用上，同时在主线程的 IO 多路复用上阻塞等待 IO 事件的到来，本系统将使用反应堆模型和线程池来设计服务器[2]。

2．4 IO 多路复用 epoll

IO 多路复用指的是单线程或单进程同时监测多个文件描述符是否可执行 IO 操作能力，由于 socket 的起源是 Unix，在 Unix/Linux 中的一个基础概念就是一切皆是文件，都可以打开(Open)然后进行读(read)写(write)到关闭(close)，在创建 socket 时是会产生一个文件描述符通过该文件描述符进行数据之间的读写，为了在有限的资源下提高并发的效率，因此，可以使用 IO 多路复用的方式去解决 socket 的并发问题，利用 IO 多路复用去实现一个服务器可以响应多个客户端的事件。

3 系统流程设计

3.1 用户功能设计

若用户没用账号则需要进行注册，前往注册窗口进行用户的注册，用户填写好名称与密码等信息后系统会将信息发送给服务器，服务器会随机生成账号并判断随机账号是否已经存在如果未存在则会将用户的信息和账号绑定后存入数据库，如果随机生成账号已经存在则重新生成再次进行判断，成功存入数据库后会将账号发送回给客户端。

用户就可以输入服务器提供的账号进行登录，登录时将信息发送给服务器，服务器会进行一个判断，判断账号和密码是否与服务器数据库内的账号密码匹配，如果匹配则会发送登录成功信息给客户端，如果未找到账号或者密码不正确都不能登录成功，登录失败会将登录失败信息发回给客户端。

3.2 好友和群聊功能

用户可以通过好友搜索功能向对方发送添加好友请求，对方可以同意和拒绝，对方同意后可以对好友进行删除和聊天[3]。

用户可以通过创建群聊功能，对群聊进行增删改查，同时还可以删除指定的群成员。

用户是群管理员，还也可以在群聊中发送群聊公告。

用户可以通过搜索群聊申请加入群聊，群管理员同意以后可以加入群聊进行群聊天，同时还可以退出群聊天，退出群聊的用户会给管理员发送退出消息。

管理员删除群聊后，群成员可以收到群聊解散消息。

3.3 E-R 图

E-R 图可以快速的了解和深入系统中用户与用户之间的关系，系统总体如图 3-1 所示。

4 核心模块实现

4.1 数据库的连接

连接数据库使用的 MySQL 官方所以提供的库，首先子在 Linux 环境下安装 MySQL 使用 sudo apt-get install libmysqld-dev 命令下载对应的 MySQL 动态库环境，然后使用头文件 mysql.h，通过 mysql_init()函数初始化连接，再通过 mysql_real_connect()函数建立一个到 mysql 的连接，最后就可以通过 mysql_query()函数使用 mysql 语句进行数据库的操作。

4.2 登录功能模块

登录功能模块主要的分为客户端是实现和服务器端业务实现，客户端实现是通过 Qt 框架提供的信号和槽机制实现[6]，封装一个登录信号当用户点击登录按钮时发送登录信号，调用登录的槽函数，其中槽函数内的函数方法是先与服务器建立 socket 连接如果连接失败则无法发送消息，如果连接成功，再通过调用 Linux 系统提供的 C 语言函数 write()函数进行 socket 数据发送，将结构体数据类型转化成 const char 指针后发送给服务器端[7]。

服务器端服务器在启动时就先将数据库连接池启动，再通过 Linux 提供的 read 函数收到 const char 类型的数据[8]，首先解析一个 int 长度的消息标识符判断是登录消息后进入系统的登录函数对象[9][10]，在函数对象中调用数据库连接池类中的 SqlQuery 函数来对账号密码在数据库中进行匹配，如果匹配正确则通过 write 函数回发给客户端登录成功和基本的用户信息

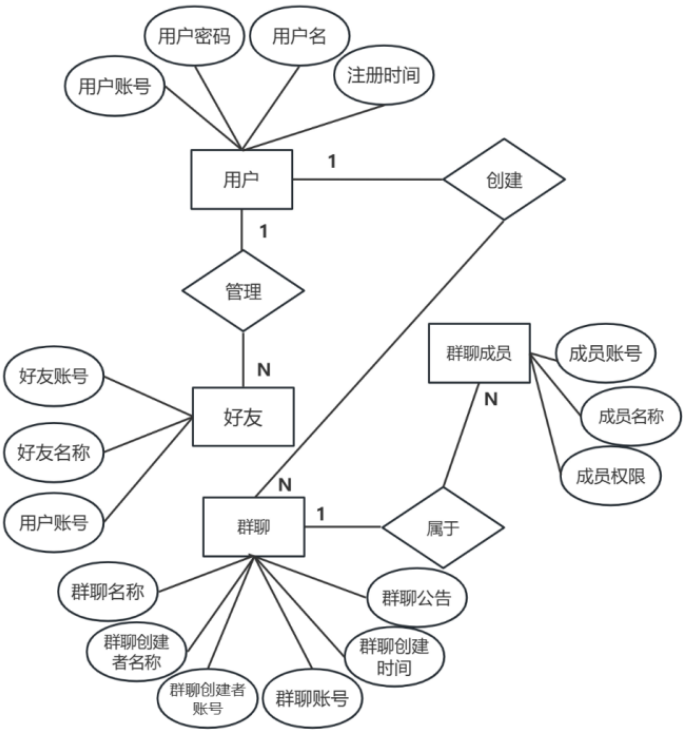

图 3-1 系统总体平台 E-R 图

并将用户的 socket 连接保存到用户在线类 OnlineMap 中。

4.3 好友功能模块

（1） 搜索好友功能

用户通过在客户端的搜索框中输入另一个好友的账号点击搜索按钮后将好友的数据打包发送给服务器端，服务器对消息进行解析后对数据库进行搜索，搜索到好友的信息后返回给客户端。搜索好友服务器端的关键逻辑代码：

```
FindIdEcho* FindUserService::operator()(Connect* _connetion,
std::shared_ptr<Mysqlpool> _mysql, const char* _str)
{
    std::cout << "FindUserService()\n";
    FindUserIdMsg findmsg;
    FindIdEcho findecho;
    memset(&findmsg, 0, sizeof(findmsg));
    memset(&findecho, 0, sizeof(findmsg));
    memcpy(&findmsg, _str, sizeof(findmsg));
    std::string sql = "select u_id,name from im_user where
u_id='";
    sql.append(strtool::IntToString(findmsg.find_id)).append("'")
;
    MYSQL_ROW row = _mysql->SqlQuery(sql.c_str());
    if (row != NULL) {
        findecho.flag = 4;
        sscanf(row[0], "%u", &findecho.id);
        strcpy(findecho.name, row[1]);
    }
    else
    {
        findecho.flag = 4;
        findecho.id = 0;
        memcpy(findecho.name, "未找到用户", sizeof("未找
到用户"));
    }
    ptr_findecho = &findecho;
    return ptr_findecho;
}
```

（2） 添加在线好友请求

搜索出好友的结果后，通过添加好友按钮发送添加好友的信息请求，对方就可以接收到添加好友的申请，同意添加好友申请后就可以添加到服务器的数据库好友表中。添加在线好友关键逻辑代码：

```
AddFriendMsg addmsgtemp;
    memset(&addmsgtemp, 0, sizeof(addmsgtemp));
    memcpy(&addmsgtemp, _str, sizeof(addmsgtemp));
    if (addmsgtemp.addstate == 0) {
        for (auto& [k, v] : _onlinelist->getOnlineMap()){
        if (v.uid == addmsgtemp.goto_uid) {
        k->setSendBuffer((char*)&addmsgtemp,
sizeof(addmsgtemp));
        k->Write();}}}
    else if (addmsgtemp.addstate == 1) {
        char name[20] = "";
        std::string ssql = "select name from im_user where
u_id='";
    ssql.append(strtool::IntToString(addmsgtemp.goto_uid)).app
end("'");
        MYSQL_ROW row = _mysql->SqlQuery(ssql.c_str());
```

```
            memcpy(name, row[0], sizeof(name));
            char c_sql[256];
            sprintf(c_sql, "insert into im_friend
(uid,friend_id,friend_name) values
('%u','%u','%s'),('%u','%u','%s')",
            addmsgtemp.from_uid, addmsgtemp.goto_uid, name,
addmsgtemp.goto_uid, addmsgtemp.from_uid,
addmsgtemp.from_name);
            std::cout << "sql:" << c_sql << "\n";
            bool ret = _mysql->SqlInsert(c_sql);
            if (ret) {
                  for (auto& [k, v] : _onlinelist->getOnlineMap()){
                        if (v.uid == addmsgtemp.goto_uid)
{k->setSendBuffer((char*)&addmsgtemp, sizeof(addmsgtemp));
      k->Write();}}}else {}}
      else if (addmsgtemp.addstate == 2){for (auto& [k, v] :
_onlinelist->getOnlineMap()){
                  if (v.uid == addmsgtemp.goto_uid) {
                  k->setSendBuffer((char*)&addmsgtemp,
sizeof(addmsgtemp));
      k->Write();}}}
```

4.4 好友在线聊天功能

好友聊天模块，用户打开好友列表，选着好友进行聊天，发送消息后服务器会判断好友是否在线，如果好友不在线则不发送消息，如果好友在线好友就回收到用户发送的聊天消息，并在聊天记录显示用户发送来的消息。好友在线聊天功能关键代码如下：

```
void ChatService::operator()(Connect* _conn, OnlineMap*
_online, Mysqlpool* _mysql, const char* _str)
{
      memset(&chatMsg, 0, sizeof(chatMsg));
      std::string _onlineChatSql = "insert into im_chat_online
(flag,from_uid,texts,goto_uid,msg_state,texts_time) values('";
      memcpy(&chatMsg, _str, sizeof(chatMsg));
      for (const auto& [conn, onlineUser] :
_online->getOnlineMap())
      {
            if (onlineUser.uid == chatMsg.goto_uid)
            {
                  conn->setSendBuffer((char*)&chatMsg,
sizeof(chatMsg));
                  conn->Write();
            }
      }
}
```

4.5 删除好友功能

删除好友模块，客户端右键选择好友组件弹出删除菜单，当用户点击删除并确认删除时，发出删除信息到服务器，服务器对消息解析判断是删除好友消息后，调用数据库里连接池类中的函数 SqlInsert 插入两条删除语句将好友和用户将数据库好友表中删除。

```
void DeleteService::DeleteFriend(DeleteMessage* msg)
{
      char sql1[MAX_SQL_LEN_3] = "";
      sprintf(sql1, "delete from im_friend where uid='%u' and
friend_id='%u'", msg->from_id, msg->del_id);
      char sql2[MAX_SQL_LEN_3] = "";
      sprintf(sql2, "delete from im_friend where uid='%u' and
friend_id='%u'", msg->del_id, msg->from_id);
      bool ret1 = this->_mysql->SqlInsert(sql1);
      bool ret2 = this->_mysql->SqlInsert(sql2);
      if (ret1 && ret1) {
            std::cout << "删除成功!!\n";
            for (auto& [k, v] : _online->getOnlineMap())
            {
                  if (v.uid == msg->del_id) {
                        k->setSendBuffer((char*)msg,
sizeof(DeleteMessage));
                        k->Write();
                  }
            }
      }
}
```

4.6 群聊模块

（1）创建群聊

操作员点击周报对应的预览按钮即可预览自己所创建群聊功能的实现，用户在客户端输入群聊的基本信息，群聊账号，群聊名称，群聊简介。

（2）删除群聊

删除群聊功能，用户在客户端群聊列表中选择群聊打开群聊窗口，群聊窗口通过 read_group_info 函数事先加载群的信息如群成员列表等，用户打开窗口后会对打开窗口用户进行判断，如果用户是该群的创建者着出现解散群聊按钮。

（3）搜索群聊

搜索群聊功能，客户端功能是通过用户输入搜索的群聊账号后点击按钮发送一个搜索的消息到服务器端中，服务器对消息进行解析后去数据库中搜素群聊，如果群聊存在将该群聊信息回显给客户端，在客户端中进行显示。

（4）加入群聊

入群聊模块，用户打开搜索窗口输入要加入的群聊账号，如果已经是群聊成员则会提示已经是群聊成员，否则会发送到服务器进行搜索，如果群聊存在则返回群聊的基本信息，用户可以申请加入群聊如果管理员在线则会向管理员发送申请加入群聊信息。

（5）退出群聊

退出群聊模块，用户打开群聊如果用户是该群的普通用户，则可以点击退出群聊按钮，如果确认退出会想服务器发送消息，通知该群的所有成员用户退出了群聊，系统会将用户从群聊列表中移除。

5 结束语

系统采用的是 C++作为主要开发语言，围绕着 Linux 系统下的 Socket 进行一个封装，利用 epoll 加上多线程为先关技术，通过反应堆模式设计服务器端，通过主要的 mainReactor 来为服务器与客户端建立新的连接，再通过 subReactor 基于 mainReactor 注册的 Socket 多路分离 IO 读写事件这部分交给多线程，转化过来就是当用户登录的时候有主要的 Reactor 来完成连接，再将这个连接的读写任务交给线程池去完成数据的读写操作。

系统的客户端采用的是 Qt 作为客户端图形化界面的开发，极大的缩短了客户端的开发时间，通过利用 Linux 提供的 Socket 与服务器进行通信，将界面的信息转化成结构体再讲结构体转化成字符串进行一个网络数据的传输，使用的是 TCP 的协议方式进行连接，TCP 作为面向连接的一种协议对比于 UDP 相对来说比较连接的可靠。

参 考 文 献

[1] 陈宁. 基于 TCP/IP 协议的 PLC 远程无线通讯系统研究[D]. 河北:河北科技大学,2019.

[2] 陈亮平,罗南超. Linux 下基于 Epoll+线程池简单 Web 服务器实现[J]. 福建脑,2019,35(4):24-27. DOI:10.16707/j.cnki.fjpc.2019.04.006.

[3] 王林. 基于 C/S 模式的高并发局域网聊天系统设计[D]. 安徽:合肥工业大学,2020.

[4] 李明,陈琳. Linux 下高并发服务器的研究与实现[J]. 电脑知识与技术,2019,15(23):259-261.

[5] 王维,陈伟,聂维. 基于 Linux 聊天系统的设计与实现[J]. 数字技术与应用,2017(10):154-155.

[6] 张劲峰. 基于 Qt 的跨平台 web 服务开发框架[D]. 陕西:西安电子科技大学,2018.

[7] 闫锋欣,牛子杰,杜烁炜,等. 基于 Qt 的 Android 应用程序 C/C++开发方法与实践[J]. 计算机系统应用,2018,27(7):96-102.

[8] 严谦,阳泳. 网络编程 tcp/ip 协议与 socket 论述[J]. 电子世界,2016(8):68-68,70.

[9] 李亚朋. 多线程下负载均衡的 OpenMP 静态调度方法[D]. 河南:郑州大学,2021.

[10] 鲁辰喜. 基于背景知识的聊天系统研究与实现[D]. 四川: 电子科技大学,2021.