

四川大學



# 计算机网络和分布式系统 项目开发报告

题 目 InstantMessagingutility

学 院 建筑与环境学院

专 业 力学-软件工程实验班

学生姓名 胡方晨

学 号 2018141501263 年级 2019 级

指导教师 宋万忠

二〇二一年 06 月 16 日



## 目录

一、 功能及说明	3
二、 实现功能的流程及说明	5
三、 项目开发使用的知识点列表	9
四、 项目开发中遇到的问题和体会	9
五、 附件一：代码	见 GitHub



## 一、功能及说明

(1) 打开Linux（我默认是Ubuntu）的终端，进入 build 目录，依次执行 cmake 和 make 指令（GitHub 文件夹已经执行 cmake 和 make，如要重新生成，请先删除 build 和 bin 内文件）

```
fangchen@fangchen-virtual-machine:~/instant_messaging_utility/build$ cmake ..
-- The C compiler identification is GNU 4.8.5
-- The CXX compiler identification is GNU 4.8.5
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /home/fangchen/instant_messaging_utility/build

fangchen@fangchen-virtual-machine:~/instant_messaging_utility/build$ make
Scanning dependencies of target chatroom_client
[ 16%] Building CXX object CMakeFiles/chatroom_client.dir/src/Client.cpp.o
/home/fangchen/instant_messaging_utility/src/Client.cpp: In member function 'void Client::Start()':
/home/fangchen/instant_messaging_utility/src/Client.cpp:108:44: warning: ignoring return value of 'char* fgets(char*, int, FILE*)', declared with attribute warn_unused_result [-Wunused-result]
        fgets(message, BUF_SIZE, stdin);
        ^
[ 33%] Building CXX object CMakeFiles/chatroom_client.dir/src/ClientMain.cpp.o
[ 50%] Linking CXX executable ../bin/chatroom_client
[ 50%] Built target chatroom_client
Scanning dependencies of target chatroom_server
[ 66%] Building CXX object CMakeFiles/chatroom_server.dir/src/Server.cpp.o
[ 83%] Building CXX object CMakeFiles/chatroom_server.dir/src/ServerMain.cpp.o
[100%] Linking CXX executable ../bin/chatroom_server
[100%] Built target chatroom_server
```

(2) 然后进入 bin 目录，首先运行 chatroom\_server

```
fangchen@fangchen-virtual-machine:~/instant_messaging_utility/bin$ ./chatroom_server
Init Server...
Start to listen: 127.0.0.1
fd added to epoll!
```

(3) 之后打开新的终端运行 chatroom\_client



```
fangchen@fangchen-virtual-machine:~/instant_messaging_utility/bin$ ./chatroom_client
Connect Server: 127.0.0.1 : 8888
fd added to epoll!

fd added to epoll!

Please input 'exit' to exit the chat room
Welcome you join to the chat room! Your chat ID is: Client #5
```

(4) 可以看到服务端和客户端分别有一些日志输出，客户端也会收到欢迎信息。为了加入更多的客户，可以打开新的终端，启动新的客户，每个客户的 clientfd 是不同的，发出去的消息在其他客户界面都可以看到来源。

(5) 在不同的客户端界面发送信息，结果如下

```
fangchen@fangchen-virtual-machine:~/instant_messaging_utility/bin$ ./chatroom_client
Connect Server: 127.0.0.1 : 8888
fd added to epoll!

fd added to epoll!

Please input 'exit' to exit the chat room
Welcome you join to the chat room! Your chat ID is: Client #5
ClientID 6 say >> nh
ClientID 8 say >> hello
ClientID 6 say >> hello
ClientID 6 say >> 大家好
ClientID 7 say >> 你好你好

fangchen@fangchen-virtual-machine:~/instant_messaging_utility/bin$ ./chatroom_client
Connect Server: 127.0.0.1 : 8888
fd added to epoll!

fd added to epoll!

Please input 'exit' to exit the chat room
Welcome you join to the chat room! Your chat ID is: Client #6
nh
ClientID 8 say >> hello
hello
大家好
ClientID 7 say >> 你好你好

fangchen@fangchen-virtual-machine:~/instant_messaging_utility/bin$ ./chatroom_client
Connect Server: 127.0.0.1 : 8888
fd added to epoll!

fd added to epoll!

Welcome you join to the chat room! Your chat ID is: Client #7
Please input 'exit' to exit the chat room
ClientID 8 say >> hello
ClientID 6 say >> hello
ClientID 6 say >> 大家好
你好你好
```



## (6) 输入 exit 推出聊天室

The screenshot shows two terminal windows. The left window is the server process, and the right window is the client process.

**Server Terminal:**

```
fangchen@fangchen-virtual-machine: ~/instant_messaging_...
client connection from: 127.0.0.1:54178, clientfd = 8
fd added to epoll!
Add new clientfd = 8 to epoll
Now there are 4 clients in the chat room
welcome message
epoll_events_count = 1
read from client(clientID = 8)
epoll_events_count = 1
read from client(clientID = 6)
epoll_events_count = 1
read from client(clientID = 6)
epoll_events_count = 1
read from client(clientID = 7)
epoll_events_count = 1
read from client(clientID = 8)
ClientID = 8 closed.
now there are 3 client in the chat room
```

**Client Terminal:**

```
fangchen@fangchen-virtual-machine: ~/instant_messaging_utility/bin$ ./chatroom_client
Connect Server: 127.0.0.1 : 8888
fd added to epoll!

Welcome you join to the chat room! Your chat ID is: Client #8
Please input 'exit' to exit the chat room
hello
ClientID 6 say >> hello
ClientID 6 say >> 大家好
ClientID 7 say >> 你好你好
exit
fangchen@fangchen-virtual-machine: ~/instant_messaging_utility/bin$ a
```

## 二、实现功能的流程及说明

### (1) 需求分析

这个聊天室软件需要下面两个程序:

1. 服务器: 能够接受新的客户端连接, 并将每个客户端发过来的消息发给所有其他的客户端
2. 客户端: 能够连接服务器, 并向服务器发送消息, 同时接收服务器发过来的任何消息

根据上面的需求分析, 设计所需的类。

其中客户端类我们需要支持下面几个功能:

1. 连接服务器
2. 支持用户输入聊天消息, 发送消息给服务器
3. 接收并显示服务器的消息
4. 退出连接

针对上述需求, 客户端的实现需要两个进程分别支持下面的功能:

子进程的功能:

1. 等待用户输入聊天信息



2.将聊天信息写到管道（pipe），并发送给父进程

#### 父进程的功能：

1. 使用 epoll 机制接受服务端发来的信息，并显示给用户，使用户看到其他用户的聊天信息

2.将子进程发给的聊天信息从管道（pipe）中读取，并发送给服务端

#### 服务端类需要支持：

1.支持多个客户端接入，实现聊天室基本功能

2.启动服务建立监听端口等待客户端连接

3.使用 epoll 机制实现并发，增加效率

4.客户端连接时发送欢迎消息并存储连接记录

5.客户端发送消息时广播给其他所有客户端

6. 客户端请求退出时对连接信息进行清理

## （2）程序结构

构建基本项目框架，依次建立 bin、build、lib、include、sec 和 CMakeList.txt

每个文件的作用：

1.Common.h：公共头文件，包含所需的所有宏定义及 socket 网络编程头文件

2.Client.h， Client.cpp：客户端类实现。

3.Server.h， Server.cpp：服务端类实现。

4.ClientMain.cpp， ServerMain.cpp：客户端及服务端的主函数。

头文件新建在 include， cpp 文件建立在 src 中

## （3）实现需要的类

Common. h 在这个项目中，我们只需要定义一个单独的函数被类成员函数调用即可，这个功能函数的作用就是将文件描述符 fd 添加到 epollfd 标示的内核事件表中。因此我们将函数



定义放在头文件 `Common.h` 中。除了这个功能函数之外，我们还需要把客户端和服务端共用的宏定义放在 `Common.h` 中，例如：服务器地址、服务器端口号、消息缓存大小、服务器端默认的欢迎及退出消息。

服务端类根据分析，我们需要下面的接口：初始化 `Init()`、关闭服务 `Close()`、启动服务 `Start()`、广播消息给所有客户端 `SendBroadcastMessage()`

服务器主循环中都每次都会检查并处理 `EPOLL` 中的就绪事件。而就绪事件列表主要是两种类型：**\*\*新连接或新消息\*\***。服务器会依次从这个列表中提取事件进行处理，如果是新连接则 `accept()` 接受连接并 `addfd()`。如果是新消息则广播给当前连接到服务器的所有客户端，从而实现聊天室的效果。

广播消息的代码中首先使用 `recv()` 读取收到的消息，然后查看消息长度，如果消息长度为 0，则认为是客户端中止连接的消息，从而 `close()` 并从客户端列表中移除该客户端。如果消息长度不为 0 则为有效的消息，需要首先 `sprintf()` 对消息进行格式化，包含一些必要的信息，然后从客户端列表中循环取出每个客户端的 `fd`，使用 `send()` 发出消息给每个客户端。

`Server.h` 及 `Server.cpp` 文件实现后，我们需要完成 `ServerMain.cpp` 文件中的主函数。主函数只需要创建一个 `Server` 对象，并调用 `Start()` 接口。

客户端类根据分析，我们需要下面的接口：连接服务端 `Connect()`、退出连接 `Close()`、启动客户端 `Start()`

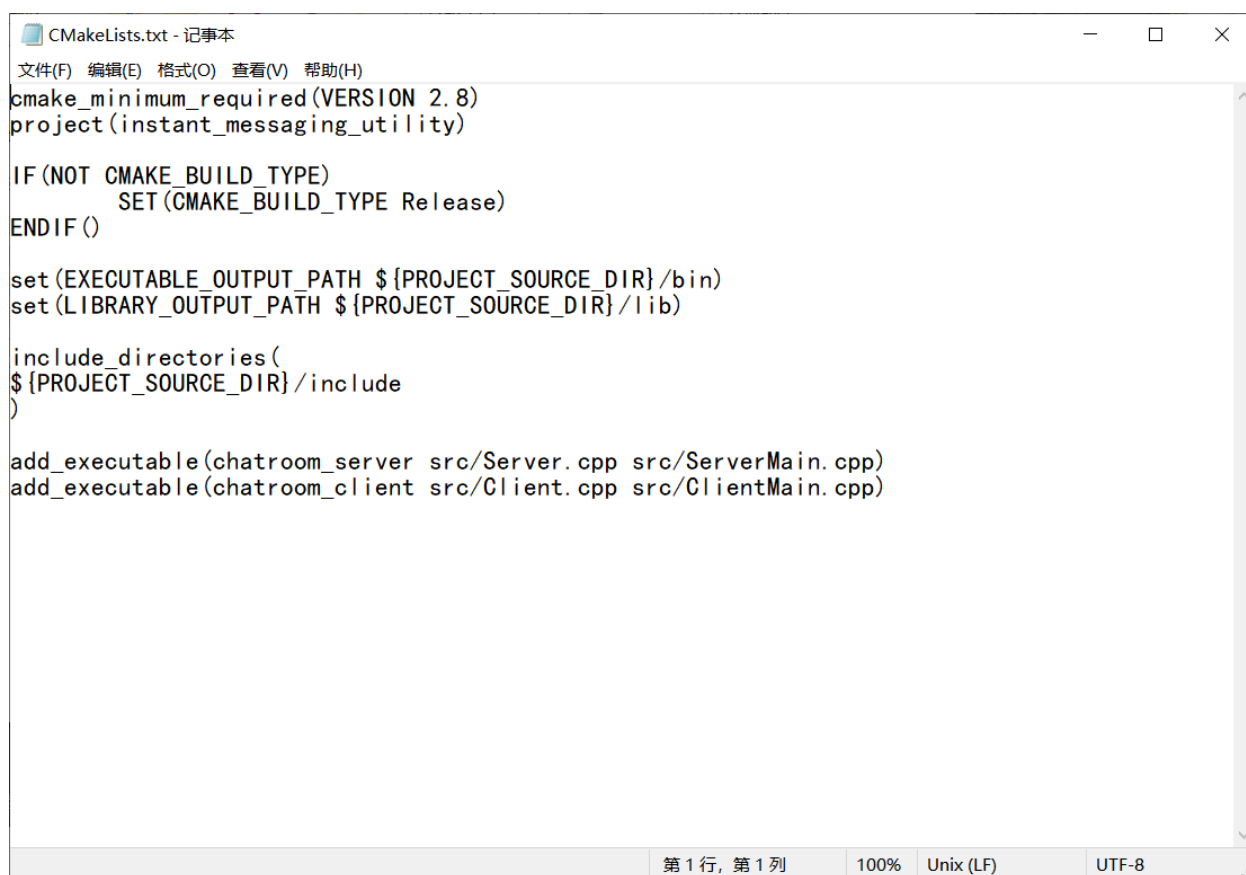
`Client.h` 及 `Client.cpp` 文件实现后，我们需要完成 `ClientMain.cpp` 文件中的主函数。主函数只需要创建一个 `Client` 对象，并调用 `Start()` 接口。

#### (4) 编写 `CMakeLists.txt`、编译及运行



一般开头都是给出要求 `cmake` 最低版本以及工程名称，然后就是可将编译方式设置为 `Release`，该模式对应的模式为 `debug`。后者是调试模式，系统运行会慢很多。所以要求有比较高效的运行方式，建议选择 `Release`。然后设置可执行文件与链接库保存的路径。然后设置头文件目录使得系统可以找到对应的头文件。然后选择需要编译的源文件，凡是要编译的源文件都需要列举出来。

完整的 `CMakeLists.txt` 文件内容如下图所示：



```
cmake_minimum_required(VERSION 2.8)
project(instant_messaging_utility)

IF(NOT CMAKE_BUILD_TYPE)
    SET(CMAKE_BUILD_TYPE Release)
ENDIF()

set(EXECUTABLE_OUTPUT_PATH ${PROJECT_SOURCE_DIR}/bin)
set(LIBRARY_OUTPUT_PATH ${PROJECT_SOURCE_DIR}/lib)

include_directories(
    ${PROJECT_SOURCE_DIR}/include
)

add_executable(chatroom_server src/Server.cpp src/ServerMain.cpp)
add_executable(chatroom_client src/Client.cpp src/ClientMain.cpp)
```

打开 Linux（我默认是 Ubuntu）的终端，进入 `build` 目录，依次执行 `cmake` 和 `make` 指令（GitHub 文件夹已经执行 `cmake` 和 `make`，如要重新生成，请先删除 `build` 和 `bin` 内文件）。此时 `bin` 目录下有编译完成的 `chatroom_server` 和 `chatroom_client`。





## (5) 测试

见一、功能与说明

## 三、项目开发使用的知识点列表

C++语言基本语法

基本的 CMakeLists

C++面向对象程序设计

epoll 网络编程

计算机网络基本知识

## 四、项目开发中遇到的问题和体会

CMakeLists 知识不会要在课后学习。服务器有时异常关闭，端口还没有释放，可以修改

Common.h 中使用的服务器端口换一个编译继续使用。

体会：计算机网络知识在现代互联网体系中及其重要。

## 五、附件一：代码

见 GitHub 文件夹，主要是 src 文件