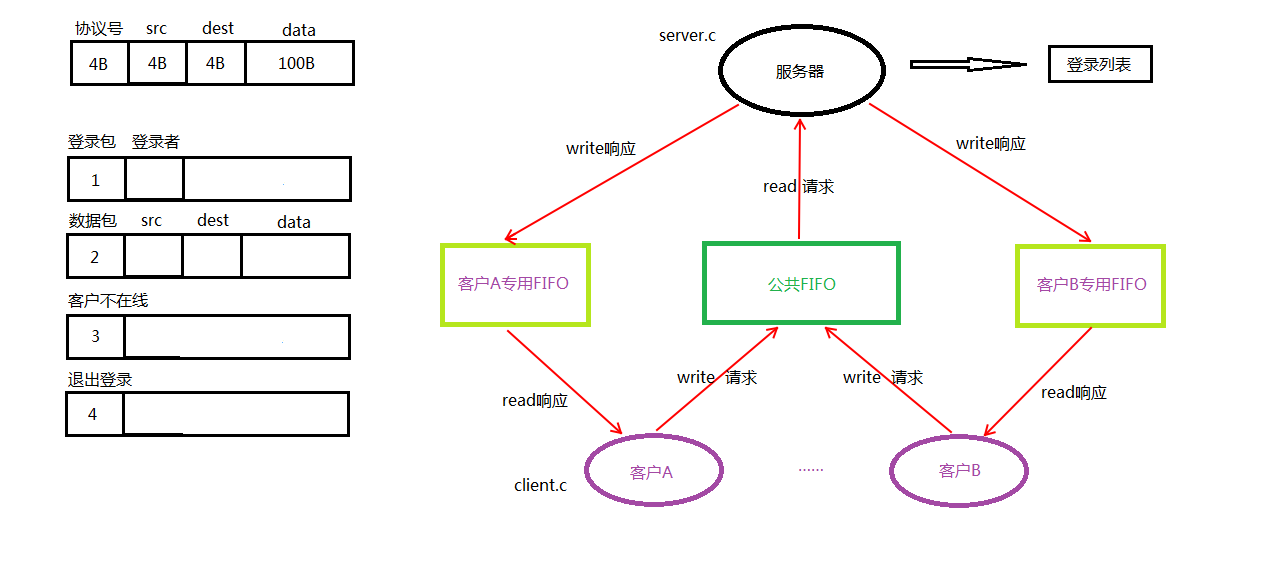
**程序：**利用FIFO实现本地聊天室

**程序功能：**实现本地进程间的相互通信

**程序分析：**通过FIFO管道可以实现本地（本机）上无血缘关系（进程不是fork()出来的）的进程间的通讯，而FIFO管道支持“一写多读”，“多写一读”。我们可以利用这一特点建立本地聊天室的模型。

服务器提供一条公共的管道，各个客户端通过这条公共管道发送消息给服务器，服务器接收到消息后，对消息包进行解析后交给相应的函数处理。如果是一个客户端发给另一个客户端的消息，就将消息通过私有管道转发下去，交给对应的客户端。



**程序种使用的知识点：**

1. 临时文件（unlink）：因为通信时需要给每个客户端都创建一条私有的通道，如果使用普通的方式创建文件，就会产生大量的管道文件。但是如果使用数据库，就可以创建普通的文件，因为管道是根据客户端的客户名创建的，如果每次输入的名字的不同就产生不同的文件，使用数据库，可以通过读入客户机的文件名来解决这个问题。
2. 修该文件属性（fcntl）：无论是客户端还是服务器都要求发送消息的同时，可以接收消息，这就要求文件是非阻塞的。使得文件获得非阻塞的属性的方法有两种：
3. 在文件打开时，将文件修改为非阻塞的（私有管道）。
4. 在文件打开后，通过fcntl函数将文件的属性修改为非阻塞

STDIN\_FILENO会在程序运行时，默认打开！

（3）字符串处理：在客户端相互发送消息时，需要输入接收方的客户机名称，规定格式为[接收方姓名:消息内容]。客户会输入一串字符串，需要进程处理，将字符串以第一个“:”为分割符，分割为两部分。如果没有按格式输入则会将消息群发（发给客户端列表上的每一个客户端）。

（4）深拷贝：在给字符串进程赋值一定要注意，使用strcpy函数，而不是简单的使用“=”。

1. 范型指针：将消息结构体转换为“范型指针”（void\*），然后是使用char\*接收（也可以直接使用viod\*接收）。
2. access函数：检查调用进程是否可以对指定的文件执行某种操作。

int access(const char \* pathname, int mode)

pathname:需要检测的文件路径名

mode:需要测试的操作模式

R\_OK 测试读许可权

W\_OK 测试写许可权

X\_OK 测试执行许可权

F\_OK 测试文件是否存在

成功执行时，返回0。失败返回-1并设置errno

**程序中存在的问题：**

（1）在客户端的管理上，使用了数组。数组虽然使用比较简单，但后期会有很多的麻烦，如空间浪费，客户端退出的处理不够灵活。

（2）在客户端登录时，没有对数据进程越界判断。

（3）没有对客户端的输入的客户名进行检查（会造成消息混乱）。

（4）数据包使用了定长包，处理上比较简单，但造成了浪费。

（5）客户端没有处理服务器退出的机制（虽然函数会自动关闭，但显得代码不是很完善）。

**运行环境：**

Ubuntu18.04+gcc

**使用说明与注意事项：**

1.公共管道使用mkfifo在控制台下直接建立的，然后使用了宏定义和绝对路径。

#define SERVER\_FIFO

"/home/dragon/Documents/study/3\_1Day\_IPC/hw/qq\_local/ff1"

默认私有管道的名称为客户机名称，创建在当前目录下，临时文件，使用后会被自动删除。

2.客户端和服务器的退出命令都为：qquit

客户端退出时，会关闭私用管道和公共管道（它自己的那一条的写端）

服务器退出时，会给每一个在线的客户端的发送消息，然后关闭私有的管道的写端和公共管道的读端。

3.客户端输入用户名采用的是scanf，因此不能使用空格

在输入时不能输入空格，因为scanf函数遇空格会终止，会将消息从空格处截断，后面的消息会变成群发。

3.1可以利用scanf的格式控制来解决。也

3.2可以使用gets\_s函数（需要将文件改为cpp），

3.3不推荐使用gets函数（没有错误检测，容易溢出）。

3.4 可以使用fgets