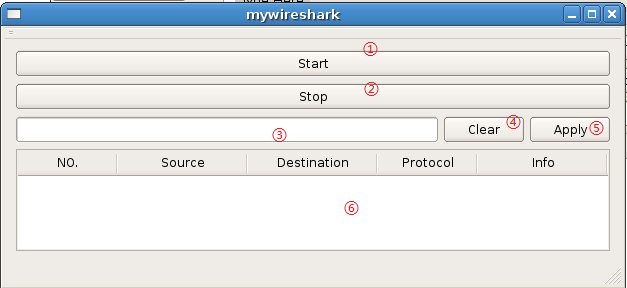
**界面说明**



①开始抓包，并在6中显示

②停止抓包

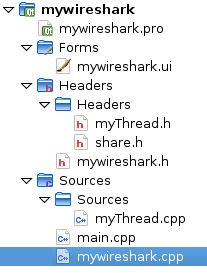
③输入筛选的条件，例如：UDP（大写字母）

④清空输入并显示所有的包

⑤应用输入的条件

⑥显示抓到的包

**文件说明**



mywireshark.ui: 窗体界面设计

myThread.h: 自定义线程类

share.h: 数据结构定义

myThread.cpp: 自定义线程运行函数

main.cpp: 主函数

mywireshark.cpp: 主线程函数，定时函数，按钮函数等

**程序说明**

**一、参数说明**

typedef struct \_rowdata //自定义结构体，用来存储抓到的包

{

int num;

char buffer[1518]; //整个包存储在buffer里

char proto[20]; //协议

char sip[100]; //源ip地址

char dip[100]; //目的ip地址

char info[100]; //其他信息

} ROW\_DATA;

int iscatch = 0; //是否抓包，是1，否0

long rowcount = 0; //将要在窗体里显示的行数

long old\_rowcount = 0; //当前窗体里显示的行数

unsigned char \*p = NULL;

QVector<ROW\_DATA> rowdata; //用QVector容器来存储所有抓到的包

QVector<ROW\_DATA> sel\_rowdata; //用QVector容器经过筛选出来的包

ROW\_DATA rowdatabuffer; //当前抓到的包

QString sel = ""; //筛选条件

**二、程序说明**

程序运行起来有两个线程，主线程负责显示界面上的信息，副线程负责抓包。

主线程里有一个定时器，执行周期为20ms；执行函数首先判断筛选条件和筛选列表是否为空，如果为空，则说明要显示所有的包，否则显示经过筛选出来的包。刷新rowcount及old\_rowcount，如果old\_rowcount < rowcount，则说明有新抓到的包没有显示在界面上。用for循环从old\_rowcount开始到rowcount结束，将未显示在界面上的包显示出来。

Clear按钮将清空输入行，界面上显示的数据，经过筛选存储在sel\_rowdata里的包和筛选条件sel。Apply按钮将设置筛选条件sel，清空界面上显示的数据和经过筛选存储在sel\_rowdata里的包，然后遍历rowdata里面存储的所有包，将符合筛选条件的包复制到sel\_rowdata里。

副线程创建一个socket，然后执行死循环。死循环里首先判断iscatch是否为真，为真则用recvfrom抓包，把抓到的包保存在rowdatabuffer.buffer里。然后分析这个包所使用的协议，源ip地址，目的ip地址，分别存储到rowdatabuffer.proto, rowdatabuffer.sip, rowdatabuffer.dip。然后将整个rowdatabuffer添加到rowdata容器里。最后判断这个包的协议是否符合筛选条件，如果是，则同时添加到sel\_rowdata里。