Test.cpp

int main() ---> test is the main entry of c++ program

{

CRSDLite rsdLite;

CSDLClient sdlClient;

printf("Starting Client\n");

if (!sdlClient.Start())

{

return -1;

}

printf("Blocking on Client\n");

sdlClient.StartClient((void\*)&rsdLite); // CRSDLite rsdLite;

sdlClient.Stop();

printf("Exiting test\n");

return 0;

}

Between JAVA GUI RSDLite and test->CSDLClient

main()

CSDLClient::Start

|\_CSDLSocket::Connect("localhost", 3501);

From JAVA -> C++

main()

|\_CSDLClient::StartClient - while cycle

|\_msgId = CSDLSocket::ReadInt();

data:CSDLSocket::ReadInt

|\_CSDLSocket::ReadInts

|\_ CSDLSocket::Read

|\_select & recv

CSDLSocket::ReadString

|\_CSDLSocket::Read

|\_select & recv

From C++ -> JAVA

CRSDLite::SendPacket

|\_CSDLClient::SendPacket

according to different msg id

call different CSDLSocket::WriteInt or CSDLSocket::WriteString

|\_CSDLSocket::Write

|\_send (m\_socket

CRSDLite::OnFlashMessage

|\_CRSDLite::SendPacket

libRSDLite.so

g\_nPRCQueueID = msgget(iKey, IPC\_CREAT | IPC\_EXCL| 0600);

thread-1:CRSDLite::ProcMsgHandlerThrd

a do-while cycle to use message queue to receive messages sent from APP like - CMultiFlashFlex::StartMultiFlashFlexThread

and update GUI status

g\_nFWQueueID -> created by PST and return by calling PSTInitializeEx

thread-2:

CRSDLite::FWMsgHandlerThrd

a do-while cycle to use msg queue to receive messages from PST

there are two major msgs:

WM\_PST\_DEVICE\_ARRIVAL

WM\_PST\_DEVICE\_REMOVAL

线程模型

模拟CEvent

g\_evtStartPRCThread.Lock();

g\_evtEndPRCThread.SetEvent();

线程间同步技术：

1） 线程间通讯 消息队列

msgget();msgsnd();msgrcv();msgctl();

<http://blog.csdn.net/ta893115871/article/details/7506403>

2) 线程同步：

互斥量+条件变量

pthread\_mutex\_t

PTHREAD\_MUTEX\_RECURSIVE

如果一个线程对这种类型的互斥锁重复上锁，不会引起死锁，一个线程对这类互斥锁的多次重复上锁必须由这个线程来重复相同数量的解锁，这样才能解开这个互斥锁，别的线程才能得到这个互斥锁。如果试图解锁一个由别的线程锁定的互斥锁将会返回一个错误代码。如果一个线程试图解锁已经被解锁的互斥锁也将会返回一个错误代码。这种类型的互斥锁只能是进程私有的（作用域属性为PTHREAD\_PROCESS\_PRIVATE）。

复习一下：

[临界区(Critical section)与互斥体(Mutex)的区别](http://blog.csdn.net/fisher_jiang/article/details/2475788)

1、临界区只能用于对象在同一进程里线程间的互斥访问；互斥体可以用于对象进程间或线程间的互斥访问。  
2、临界区是非内核对象，只在用户态进行锁操作，速度快；互斥体是内核对象，在核心态进行锁操作，速度慢。  
3、临界区和互斥体在Windows平台都下可用；Linux下只有互斥体可用。

设备检测模型

CRSDLite::Initialize

|\_PST\_IPC\_KEY iTemp;

if (PST\_OK != PSTInitializeEx(MODE\_USB, &iTemp)) -> output the iTemp and set to g\_nFWQueueID

|\_PSTRegisterForNotifications

|\_CPSTMsgDispatcher::GetInstance()->RegisterIPC(ipd\_id);

|\_msgget create msg queue to share msg channel between libRSDLite and PST

the msg queue handle will be stored in m\_mapClients and PST will use it to dispatch/send msg to upper libRSDLite

|\_(called in PSTInitializeEx) CPSTSessionMgr::Initialize

|\_CPSTSession::Initialize();

| |\_pthread\_create - CPSTSession::DeviceChangeNotice

|\_m\_pInstance->m\_pUSBNotify = new CPSTUSBNotify();

CPSTUSBNotify::CPSTUSBNotify()

|\_CPSTUSBNotify::RegisterNotification

|\_pthread\_create - UdevMessageRcv

thread-1:

CPSTSession::DeviceChangeNotice

DeviceChangeNotice is a do-while to try to receive device notification message from another thread-2.

thread-2

UdevMessageRcv, in PSTUSBNotify.cpp

listens to UDEV Messages

create a msg que as QUEUE\_ID and wait for it in a do-while

thread-1 and thread-2 share a queue

class CPSTSession

{

queue<DeviceData\*> m\_DeviceQue; //hold the run queue

}