1， 基本线程

webretc中主要有两个线程：

signaling\_thread-->处理小工作量方法。要求此线程内的方法都必须快速返回。

worker\_thread-->处理大工作量的方法。此线程内的方法可能会处理很长时间，如网络消息等。

2， 线程实现

webrtc中的线程包含了一个消息队列。当需要在此线程中运行逻辑时，仅需要向线程对象发送一个消息即可。其中消息包含消息的处理类。

图示

描述已自动生成

thread主要方法：

virtual void Post(MessageHandler \*phandler, uint32 id = 0,  MessageData \*pdata = NULL, bool time\_sensitive = false);

发送一个异步请求，调用会立即返回。

virtual void PostDelayed(int cmsDelay, MessageHandler \*phandler, uint32 id = 0, MessageData \*pdata = NULL)；

发送一个异步请求，调用会立即返回，但是此请求会在cmsDelay后被处理。相当于集成了个计时器。

virtual void Send(MessageHandler \*phandler, uint32 id = 0,  MessageData \*pdata = NULL);  
发送一个同步请求，消息处理完成后调用才会返回。

3， 线程调用实现

3.1 发送一个消息。

thread->send()

thread->post()

3.2 调用Invoke

worker\_thread\_->Invoke<bool>(Bind(&MediaEngineInterface::SetOutputVolume, media\_engine\_, level));

以上调用表示在worker\_thread\_内同步调用media\_engine\_类的MediaEngineInterface::SetOutputVolume方法，方法有一个参数是level

3.3 用proxy宏

如下用PROXY宏定义对应类的proxy类，此proxy类的所有方法都会在woker\_thread\_线程内调用

BEGIN\_PROXY\_MAP(PeerConnection)  
  PROXY\_METHOD0(talk\_base::scoped\_refptr<StreamCollectionInterface>,  
                local\_streams)  
  PROXY\_METHOD0(talk\_base::scoped\_refptr<StreamCollectionInterface>,  
                remote\_streams)  
  PROXY\_METHOD2(bool, AddStream, MediaStreamInterface\*,  
                const MediaConstraintsInterface\*)  
  PROXY\_METHOD1(void, RemoveStream, MediaStreamInterface\*)  
  PROXY\_METHOD1(talk\_base::scoped\_refptr<DtmfSenderInterface>,  
                CreateDtmfSender, AudioTrackInterface\*)  
  PROXY\_METHOD2(bool, GetStats, StatsObserver\*, MediaStreamTrackInterface\*)  
  PROXY\_METHOD2(talk\_base::scoped\_refptr<DataChannelInterface>,  
                CreateDataChannel, const std::string&, const DataChannelInit\*)  
  PROXY\_CONSTMETHOD0(const SessionDescriptionInterface\*, local\_description)  
  PROXY\_CONSTMETHOD0(const SessionDescriptionInterface\*, remote\_description)  
  PROXY\_METHOD2(void, CreateOffer, CreateSessionDescriptionObserver\*,  
                const MediaConstraintsInterface\*)  
  PROXY\_METHOD2(void, CreateAnswer, CreateSessionDescriptionObserver\*,  
                const MediaConstraintsInterface\*)  
  PROXY\_METHOD2(void, SetLocalDescription, SetSessionDescriptionObserver\*,  
                SessionDescriptionInterface\*)  
  PROXY\_METHOD2(void, SetRemoteDescription, SetSessionDescriptionObserver\*,  
                SessionDescriptionInterface\*)  
  PROXY\_METHOD2(bool, UpdateIce, const IceServers&,  
                const MediaConstraintsInterface\*)  
  PROXY\_METHOD1(bool, AddIceCandidate, const IceCandidateInterface\*)  
  PROXY\_METHOD0(SignalingState, signaling\_state)  
  PROXY\_METHOD0(IceState, ice\_state)  
  PROXY\_METHOD0(IceConnectionState, ice\_connection\_state)  
  PROXY\_METHOD0(IceGatheringState, ice\_gathering\_state)  
  PROXY\_METHOD0(void, Close)  
END\_PROXY()

PeerConnectionProxy(woker\_thread\_, peer\_connection);