服务器API说明文档

1. Node
2. int Startup(const std::string& configure)

根据配置文件启动节点

1. int AddListener(Listener\* pListener)

增加节点监听

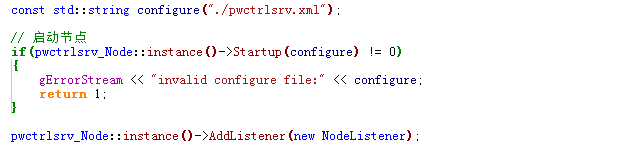


图1 启动节点，增加监听

1. int AddPort(const std::string& name,Port\* port)

增加节点线程

1. int DelPort(const std::string& name)

关闭节点线程



图2 关闭线程

1. int GetPort(const std::string& name,Port\*\*port)

获取节点线程

1. int AddRemoteNode(const std::string& name,RemoteNode\* node)

增加远程节点

1. int DelRemoteNode(const std::string& name)

删除远程节点

1. int Update()

节点更新函数

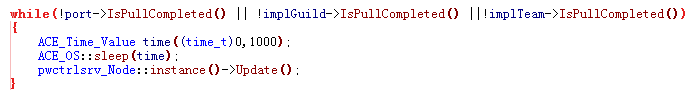


图3 节点更新

1. RemoteNode
2. int Startup(Node\* node,const std::string& addr)

远程节点启动

1. int Update(Node\* pNode)

远程节点更新函数

1. int SendMsg(int msgid,lwrpc::ProtocolObject\* object)

远程节点发送消息函数

1. Port
2. int Startup(const std::string& name,Node\* node)

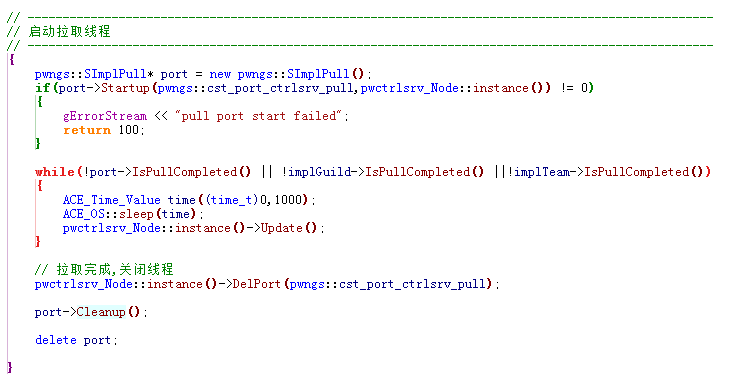


图4 线程开启/关闭

1. int Prepare2(const char\* node,const char\* port,sint64 single\_spec)

RPC调用准备

1. int Prepare(const char\* node = "",const char\* port = "",const lwrpc::sint64\* spec = NULL,int spec\_count = -1)

RPC调用准备

1. template<class P1> int Call(Obj\* obj,const char\* name,P1 p1)

RPC调用

第一种方式：同步调用

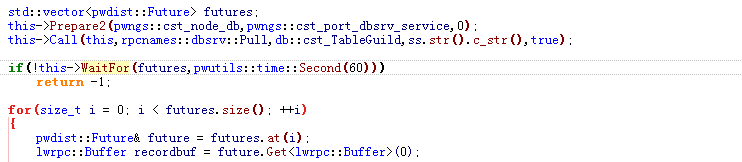


图5 RPC同步调用

第二种方式：异步调用

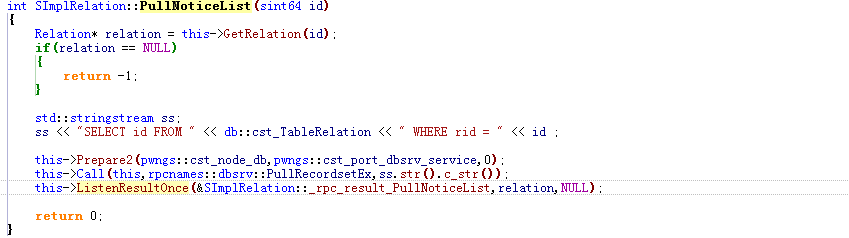


图6 RPC异步调用

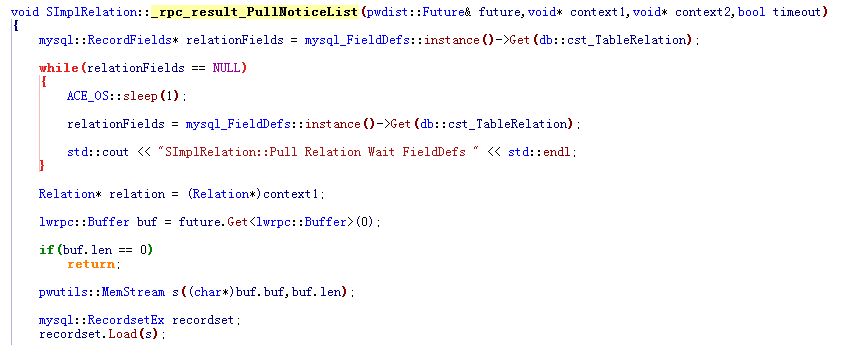


图7 RPC异步调用回调

第三种方式：函数应答，代码上下文环境自己保存，必须注册自己的返回值处理函数

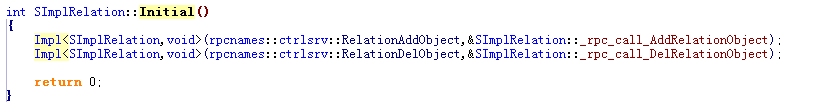


图8 注册RPC调用函数

1. RPC延后返回
2. 有序异步返回
3. 无序异步返回



图9 RPC无序调用服务端

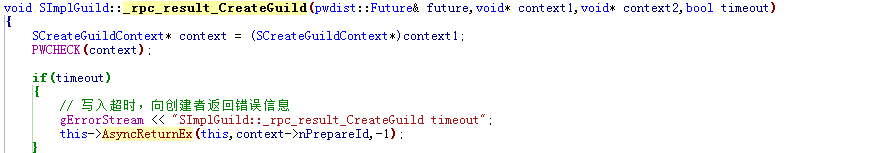


图10 返回值到客户端

1. PortEx : public Port

Lua协程同步获取

优点：易用，可将所有上层逻辑转为lua

缺点：性能稍有降低,不适用于有大量rpc调用的业务逻辑(每秒数千以上)

需求：从PortEx中继承，在Port初始化时调用LoadScriptSupport载入pwdist.lua

主要支持函数(c++):

1. pwdist\_Exec 在rpc环境中调用一个lua函数，在环境中，该lua函数可以使用所有\_\_pwdist\_打头的支持函数
2. pwdist\_ExecNonResult在rpc环境中调用一个lua函数，在环境中，该lua函数可以使用不含\_\_pwdist\_callwithresult函数在内的所有其它支持函数

以上两个函数，可将c++中的支持函数导入lua，以使lua的流程更加顺畅，使用如下：在第二个参数exports中传入SLuaExportDef结构指针，以内容为{NULL,NULL}的SLuaExportDef为结束

主要支持函数(lua):

1. \_\_pwdist\_prepare 准备一个rpc调用，最少三个参数，第三、第四个及其后，会作为c++中Prepare函数的spec参数传入
2. \_\_pwdist\_callwithresult调用远程rpc，并等待返回值，第一个参数为Obj\*，为nil表示当前Port对象，第二个参数为方法名，其后为rpc参数，成功返回多个返回值，如超时，返回nil。注：本函数返回的lwrpc::Buffer类型的数据，将在下一次\_\_pwdist\_callwithresult调用后或协程挂起后失效
3. \_\_pwdist\_call调用远程rpc，不等待返回值(有的话，该返回值会给丢弃)
4. \_\_pwdist\_return 返回值给调用方，所有参数(最多6个)都作为返回值给调用方,返回值类型限制与c++相同
5. \_\_pwdist\_log 打印log，调试用
6. \_\_pwdist\_prepare\_async\_return 准备异步返回，无参数，返回prepareid
7. \_\_pwdist\_async\_return 异步返回值,参数为prepareid + 要返回的值
8. \_\_pwdist\_clone\_buffer 复制一个lwrpc::Buffer对象，复制出来的对象可保证不会因lua协程挂起而失效

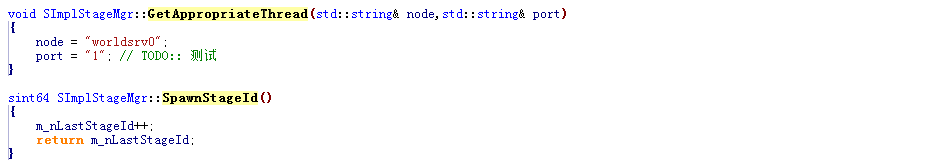




图11 C++中调用示例

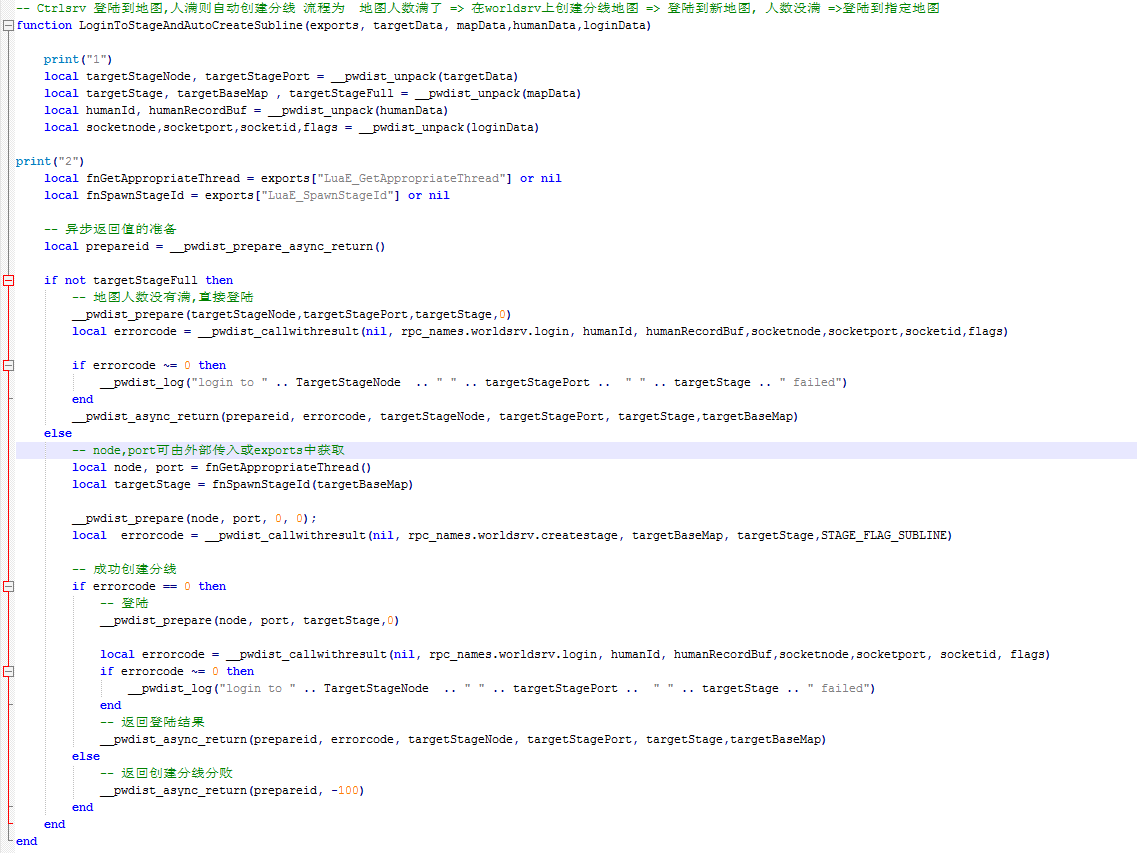


图12 lua中调用示例