**文件结构**

Source

+-Common

+-Common.h

+-FunctionMap.h

+-PackageStruct.h

+-PackageStruct\_AG.h

+-Protocol.h

+-Protocol\_AG.h

+-Utility

+-Utility.h

+-Singleton.h

+-PublicObject.h

+-PublicObject.cpp

+-MemoryFactory.h

+-MemoryFactory.cpp

+-Yond\_MD5.h

+-Yond\_XML.h

+-Yond\_XML.cpp

+-Yond\_Log.h

+-Yond\_log.cpp

+-Yond\_Thread.h

+-Yond\_Thread.h

+-Yond\_Timer.h

+-Yond\_Timer.cpp

+-Yond\_ini.h

+-Yond\_ini.h

+-Network

+-Network.h

+-Epoll\_for.h

+-Epoll\_for.cpp

+-NetworkObject.h

+-NetworkObject.cpp

+-SeesionObject.h

+-SeesionObject.cpp

+-AgentSrv

+-AgentMain.cpp

+-AgentServer.h

+-AgentServer.cpp

+-AgentPlaery.h

+-AgentPlaery.cpp

+-AgentFactory.h

+-AgentFactory.cpp

+-Handler\_FromGameServer.h

+-Handler\_FromGameServer.cpp

+-GameSrv

+-GameMain.cpp

+-GameServer.h

+-GameServer.cpp

+-GamePlaery.h

+-GamePlaery.cpp

+-GameFactory.h

+-GameFactory.cpp

+-Handler\_FromAgentServer.h

+-Handler\_FromAgentServer.cpp

+-LoginSrv

+-WorldSrv

+-DBSrv

**Utility模块说明**

2 PublicObject 公共处理类;

class PublicObject {

Public SplitString();

Public GetTimeSpanLocalToUTC();

Public GetNowTime();

Public ToTimeStampByString();

Public ToOleDateTimeString();

Public IsToday();

};

**1， MemoryFactory 内存管理；**

Template<class cType> class MemoryFactory {

Protected MemBand<cType> \* m\_pBandHead;

Protected MemBand<cType> \* m\_pFreeHead;

Protected UINT m\_dwPoolExtendSize;

Public Initialize();

Public Release();

Public Alloc();

Public Free();

};

Template<class DataType> class MemBand{

Private DataType \*\* ppDataType;

Private DataType \* pDeleteArray;

Private UInt nMaxBandObjectNum;

Private Uint FreeIdx;

Public MemBand pPrev;

Public MemBand pNext;

Public AllocBand();

Public FreeBand();

Public AllocObject();

Public FreeObject();

};

**Yond\_MD5加密功能**

**Yond\_ini配置文件解析功能**

**Yond\_Log日志处理功能**

**Yond\_Timer定时器定时功能**

**Network模块说明**

**1, Epoll\_Handler** **在实现网络模型处理时，进行回调以用于动态扩展处理；**

Epoll\_Handler在epoll\_for.h 文件实现，是基于Epoll\_For处理机制的一个回调结构体。其中包括创建连接、接收消息、发送消息、断开连接。

struct Epoll\_Handler {

void (\*CreateSocket)(socket, epoll\_head);

void (\*DoRecv)(NetworkObject \*, MSG \*);

void (\*DoSend)(NetworkObject \*);

void (\*DestroySocket)(NetworkObject \*);

};

**2, epoll\_for，实现网络模型的处理机制；**

epoll\_for 在epoll\_for.cpp文件中实现，Epoll\_for是封装Epoll的所有原理，通过Eoll\_Handler回调结构来形成扩展处理机制。

void epoll\_for(void \* param);

**3, NetworkObject 实现TCP/IP的所有网络交互功能的封装；**

class NetworkObject {

Privete socket m\_socket;

Privete struct epoll\_event m\_epollEvent;

Privete u\_int m\_IP;

Privete Short m\_Port;

Privete Buffer\* RecvBuff;

Privete Buffer\* SendBuff;

Public Init();

Public StartListen();

Public OnAccept();

Public OnSend();

Public OnRecv();

Public Connect();

Public Close();

};

**4, SessionObject 实现网络实例化和异化处理（不同的网络段不同的处理方式）；**

class SessionObject {

Privete NetworkObject \* m\_socket;

Privete NetworkType m\_intType;

Privete Bool m\_bHeartbeat;

Privete Uint m\_uiLastUpdateTime;

Privete Yond\_Timer \* m\_timerHead;

Public Init();

Public Update();

};

**AgentServer模块说明**

**核心功能说明**

**1, 实例化Epoll\_Handler，四种回调处理模型；**

void callback\_CreateSocket(socket, epoll) {

NetworkObject \* obj = new NetworkObject;

obj -> Init(socket);

obj -> AddEpollEvent(epoll);

}

void callback\_DoSend(socket, epoll, msg) {

NetworkObject \* obj = new NetworkObject;

obj -> Init(socket);

obj -> AddEpollEvent(epoll);

}

void callback\_DoRecv(socket, epoll, recvLen, msg) {

PackageStruct \*Msg = msg;

FunctionMap \* func = Find(msg->first);

AgentServer a = globle;

NetworkObject obj = a->Find(sockfd);

func(obj, msg);

}

void callback\_DestroySocket(socket, epoll, recvLen, msg) {

PackageStruct \*Msg = msg;

FunctionMap \* func = Find(msg->first);

AgentServer a = globle;

NetworkObject obj = a->Find(sockfd);

func(obj, msg);

};

**2, AgentFactory 工厂类，创建和回收所需内存，内存约束和高效管理；**

class AgentFactory{

Private MemoryFactory<NetworkObject> \*m\_NetworkObjectPool;

Private MemoryFactory<SessionObject > \*m\_NetworkObjectPool;

Private MemoryFactory<AgentPlaery> \*m\_AgentPlaeryPool;

Public Init();

Public NetworkObject \* AllocNetworkObject();

Public SessionObject \* AllocSessionObject();

Public AgentPlaery \* AllocAgentPlaery();

Public FreeNetworkObject(NetworkObject \*);

Public FreeSessionObject(SessionObject \*);

Public FreeAgentPlaery(AgentPlaery \*);

};

**3, AgentServer 是服务器root入口，所以资源从这里初始化、处理、释放；**

class AgentFactory : Single<AgentServer> {

Private Epoll\_Handler \* EH;

Private AgentFactory \* m\_cAgentFactory ;

Private Map<Uint, Uint>\* m\_mapSessions;

Private Map<Uint, Uint>\* m\_mapPlayers;

Public Init()

{

EH = new H;

EH->CreateSocket = callback\_CreateSocket;

EH->DoSend = callback\_DoSend;

EH->DoRecv = callback\_DoRecv;

EH->DestorySocket = callback\_DestorySocket;

}

Public Start() {

epoll\_for(\*EH);

}

Public AllocSession();

Public FindSession();

Public AllocSession();

Public AddPlayer();

Public FindPlayer();

Public FreePlayer();

};

**4 AgentPlaery记录玩家的网络信息，用于数据的转送操作；**

class AgentPlaery {

Private SessionObject \* m\_netClient;

Private SessionObject \* m\_netGame ;

Public Init();

Public SendToClient(MSG \* );

Public SendToGame(MSG \* );

};

**AgentMain主函数**

#define AGENG\_CONFIG\_PATH “agent.conf”

void main(int argc, char \* argv[]) {

Yond\_ini yini;

yini.Parser(AGENG\_CONFIG\_PATH);

Yond\_Log \* yLog = new Yond\_Log(yini);

yLog.Init();

yLog.Sart();

AgentServer \* root = new AgentServer(yini);

root .init();

root .Start();

}