**3D手游服务器端框架参考文档**

|  |  |
| --- | --- |
| 版本号 | version 1.0 |
| 初稿 | 2013-02-01 |
| 作者 | 毛永华（senfer） |

**文档信息**

**描述**

*(文档的简单描述)*

|  |  |
| --- | --- |
| 描述人 | 描述 |
| *毛永华（senfer）* | 介绍测试服务器模块中的各个组件的基本功能和系统执行流程 |
|  |  |

**历史版本**

*(文档的版本管理)*

|  |  |  |  |
| --- | --- | --- | --- |
| 版本号 | 日期 | 修改小结 | 修改人 |
| version 1.0 | 2013-1-25 | *初稿* | *毛永华* |
| Version1.1 | 2013-2-1 | *初稿* | *毛永华* |
|  |  |  |  |
|  |  |  |  |

**目录**

[第一章 testserver测试模块概要 2](#_Toc347475957)

[1.1 加密模块 2](#_Toc347475958)

[1.2 配置解析 2](#_Toc347475959)

[1.3 实体解析 2](#_Toc347475960)

[1.4 实体定义 4](#_Toc347475961)

[1.5 网络服务 5](#_Toc347475962)

[1.6 异常处理 6](#_Toc347475963)

[1.7 登录服务 7](#_Toc347475964)

[1.8 网络封装 8](#_Toc347475965)

[1.9 网络数据包 9](#_Toc347475966)

[1.10 rpc处理 10](#_Toc347475967)

[1.11 定时器 11](#_Toc347475968)

[1.12 类型定义模块 11](#_Toc347475969)

[1.13 工具包 12](#_Toc347475970)

[1.14 world模块 14](#_Toc347475971)

[1.15 world login模块 15](#_Toc347475972)

[1.16 global world 16](#_Toc347475973)

[第二章 模块执行流程 17](#_Toc347475974)

[2.1配置文件 17](#_Toc347475975)

[2.2 运行环境初始化 17](#_Toc347475976)

[2.3 服务启动流程 18](#_Toc347475977)

[2.4 rpc处理 19](#_Toc347475978)

# 第一章 testserver测试模块概要

## 1.1 加密模块

文档名：bitcryto.h bitcryto.cpp

模块名：加密模块

类：CbitCryto

方法：

*Public：*

描述：输入一个字符，返回加密字符

unsigned char Encode(unsigned char c);

描述：输入加密字符，返回解密字符

unsigned char Decode(unsigned char c);

描述：重置加密类索引

inline void Reset()；

## 1.2 配置解析

文档名：CCfgReader.h CCfgReader.cpp

模块名：配置文件解析模块

类：CCfgReader

方法：

*Public：*

描述：在字典项中map<string , map<string, string>> item<szSection, <szName, value>>

非空输出value，否则输出空串

string GetValue(const char\* szSection, const char\* szName);

描述：同前方法功能相同，不同处在于空串返回默认串strDefault

string GetOptValue(const char\* szSection, const char\* szName, const string& strDefault);

*Private：*

描述：读入配置文件，解析配置文件内容为字典项存储

void ScanCfgFile();

## 1.3 实体解析

文档名：defparser.h defparser.cpp

模块名：实体数据解析模块

类：CStr2IntMap

方法：

*Public：*

描述： 增加数据项map<string, uint32\_t> & vector<string> 其中uint32\_t为自增长id

void AddNewString(const string& s);

描述：输入id，返回vector<string>对应的string

const string& GetStrByInt(uint32\_t n) const;

描述：输入string， 返回map<string, uint32\_t>项id

int32\_t GetIntByStr(const string& s) const;

描述：为数据空间预分配长度大小为n的空间

void Assign(uint32\_t n);

描述：设置uid和string之间的映射关系

void SetNewString(uint32\_t uid, const string& s);

类：CDefParser

方法：

***Public****：*

描述： 输入默认文件路径，解析实体文件entities.xml和\*\*\*\*.def

void init(const char\* pszDefFilePath);

描述：输入实体name，返回实体类型TENTITYTYPE

TENTITYTYPE GetTypeId(const string& tname);

描述：输入实体类型id，返回实体类型名 string[]

const string& GetTypeName(TENTITYTYPE tid);

描述：输入实体属性name，返回实体属性数据类型VTYPE

VTYPE GetVTypeByStr(const string& s);

描述：输入实体类型name，返回实体对象定义 SEntityDef

const SEntityDef\* GetEntityDefByName(const string& );

描述：输入实体类型id，返回实体对象定义 SEntityDef

const SEntityDef\* GetEntityDefByType(TENTITYTYPE t);

描述：输入实体定义和实体属性名，返回实体属性定义 SEntityDefProperties

const \_SEntityDefProperties\* GetEntityPropDef(const SEntityDef\*, const string&);

描述：输入实体名和实体属性名，返回实体属性定义 SEntityDefProperties

const \_SEntityDefProperties\* GetEntityPropDef(const string&, const string&);

描述：输入实体类型id和方法id ，返回方法名

const string& GetMethodNameById(TENTITYTYPE t, int32\_t nFuncId);

描述：输入实体类型id和方法id，返回cell对应的方法名

const string& GetCellMethodNameById(TENTITYTYPE t, int32\_t nFuncId);

描述：输入CcfgReader对象指针，初始化数据库对象配置

void ReadDbCfg(CCfgReader\* r);

描述：返回数据库配置对象

const SDBCfg& GetDbCfg() const

***Private：***

描述：初始化数据类型名与标识之间的映射map<string, VTYPE>

void InitData();

描述：输入实体xml配置文件路径，解析实体对象的xml

void ParseEntitiesXml(const char\* pszFileName);

描述：输入def文件路径，解析def文件内容，返回SEntityDef ，失败抛出异常

SEntityDef\* ParseDef(const char\* pszDefFn);

描述：解析处理parent关系

void ParseParent();

描述：输入实体和SentityDef，解析实体所有的parent关系

void ParseEntityParent(const string& strEntity, SEntityDef\* pDef);

描述：输入子SEntityDef和父SentityDef，复制pParent配置数据到pChild

void CopyParentToEntity(SEntityDef\* pChild, const SEntityDef\* pParent);

描述：输入SentityDef和TiXmlElement， 输出实体默认属性SEntityDefProperties

\_SEntityDefProperties\* ReadProperty(SEntityDef\* pDef, TiXmlElement\* node);

描述：输入TiXmlElement，输出实体默认方法SEntityDefMethods

\_SEntityDefMethods\* ReadMethod(TiXmlElement\* node);

描述：返回对象映射属性 m\_typeStr2Int

const map<string, TENTITYTYPE>& GetDefTypes() const

## 1.4 实体定义

文档名：entity.h entity.cpp

模块名：实体属性定义模块

类：CEntityParent

方法：

***Public：***

描述： 输入pluto引用项，将实体数据保存到pluto中，成功返回true，否则返回false

bool PickleToPluto(CPluto& u) const;

描述：用输入的新数据更新实体的原有数据内容

void UpdateProps(map<string, VOBJECT\*>& new\_data)；

描述：将pluto中的中的数据从pluto中转到实体对象中，成功返回true，否则返回false

bool UnpickleFromPluto(CPluto& u);

## 1.5 网络服务

文档名：epoll\_server.h epoll\_server.cpp

模块名：epoll服务器管理模块

类：CEpollServer

方法：

***Public：***

描述： 输入服务器地址和端口，启动服务器

int StartServer(const char\* pszAddr, uint16\_t unPort);

描述：与服务器配置对象池中个服务器建立连接

int ConnectMailboxs(const char\* pszCfgFile);

描述：初始化服务器，循环监听网络服务，处理接收数据和数据发送

int Service(const char\* pszAddr, unsigned int unPort);

描述：处理发给自己的pluto，直接放入接收消息队列

inline void AddLocalRpcPluto(CPluto\* u)

***protected：***

描述：增加一个fd和mailbox到服务器map<int, CMailBox\*> m\_fds

void AddFdAndMb(int fd, CMailBox\* pmb);

描述： 上一个方法的重载，不同之处在于必须先创建mailbox在调用上一个函数

void AddFdAndMb(int fd, EFDTYPE efd, const char\* pszAddr, uint16\_t unPort);

描述：处理新的连接

virtual int HandleNewConnection(int fd);

描述：处理mailbox事件，检查mailbox连接是否错误，正常则调用下个函数

virtual int HandleMailboxEvent(int fd, uint32\_t event, CMailBox\* mb);

描述：处理socket事件，消息接受处理，若发生错误则关闭连接

virtual int HandleFdEvent(int fd, uint32\_t event, CMailBox\* mb);

描述：处理消息接收事件，消息接收分为包头和包体处理，若产生错误返回负数。

virtual int HandleMessage(int fd, CMailBox\* mb);

描述：连接超时处理，暂时是空处理

virtual int HandleTimeout();

描述：邮箱服务的重新连接

virtual int HandleMailboxReconnect();

描述：处理新的Accept连接

virtual int OnNewFdAccepted(int new\_fd, sockaddr\_in& addr);

描述：处理接收的pluto，中间调用rpc处理

virtual int HandlePluto();

描述：处理发送队列pluto中的数据发送

virtual int HandleSendPluto();

描述：处理上次没有发送完毕的数据

int HandleLeftPluto();

## 1.6 异常处理

文档名：exception.h exception.cpp

模块名：自定义异常处理

类：CException

方法：

***Public：***

描述：获取异常码

int GetCode() const

描述：获取异常信息

string GetMsg() const

***extern method：***

描述：传入异常码和异常消息

void ThrowException(int n, const string& s)；

描述：传入异常码和异常消息

void ThrowException(int n, const char\* s, ...);

文档名：logger.h logger.cpp

模块名：日志处理模块

类：CLogger

方法：

***Public：***

描述： 设置日志文件存放路径

void SetLogPath(const string& strPath);

描述：

CLogger& NewLine(int nFileType = 0);

描述：

Extern method：六种类型日志信息

描述：打印debug日志信息

void LogDebug(const char\* key, const char\* msg, ...);

描述：打印info日志信息

void LogInfo(const char\* key, const char\* msg, ...);

描述：打印warning日志信息

void LogWarning(const char\* key, const char\* msg, ...);

描述：打印error日志信息

void LogError(const char\* key, const char\* msg, ...);

描述：打印critical日志信息

void LogCritical(const char\* key, const char\* msg, ...);

描述：打印script日志信息

void LogScript(const char\* level, const char\* msg, ...);

## 1.7 登录服务

文档名：loginapp.h loginapp.cpp

模块名： 登陆服务器模块

类：CLoginappServer extend CEpollServer

方法：

***Protected：***

描述：处理pluto，中心处理调用CEpollServer::handlePluto()

int HandlePluto();

文档名：mailbox.h mailbox.cpp

模块名：邮箱信息模块

类：

方法：

***Public：***

描述： 执行与服务器的连接，并将mailbox的fd加入epoll池

int ConnectServer(int epfd);

描述：发送mailbox pluto队列中的数据

int SendAll();

描述：处理远程rpc调用，方法重载为5种。

template<typename T1>

bool RpcCall(CRpcUtil& r, pluto\_msgid\_t msg\_id, const T1& p1);

template<typename T1, typename T2>

bool RpcCall(CRpcUtil& r, pluto\_msgid\_t msg\_id, const T1& p1, const T2& p2);

template<typename T1, typename T2, typename T3>

bool RpcCall(CRpcUtil& r, pluto\_msgid\_t msg\_id, const T1& p1, const T2& p2, const T3& p3);

template<typename T1, typename T2, typename T3, typename T4>

bool RpcCall(CRpcUtil& r, pluto\_msgid\_t msg\_id, const T1& p1, const T2& p2, const T3& p3, const T4& p4);

template<typename T1, typename T2, typename T3, typename T4, typename T5>

bool RpcCall(CRpcUtil& r, pluto\_msgid\_t msg\_id, const T1& p1, const T2& p2, const T3& p3, const T4& p4, const T5& p5);

template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6>

bool RpcCall(CRpcUtil& r, pluto\_msgid\_t msg\_id, const T1& p1, const T2& p2, const T3& p3, const T4& p4, const T5& p5, const T6& p6);

## 1.8 网络封装

文档名：net\_util.h net\_util.cpp

模块名：网络连接封装模块

类：无

方法：

***Extern methods：***

描述：设置socket为非阻塞，出错返回false

bool MogoSetNonblocking(int sockfd);

描述：产生一个socket并返回fd

int MogoSocket();

描述：将socket fd与地址和端口绑定

int MogoBind(int sockfd, const char\* pszAddr, unsigned int unPort);

描述：设置socket为监听socket

int MogoListen(int sockfd, int backlog = 5);

描述：将socket与对应的服务器建立连接

int MogoConnect(int fd, const char\* pszAddr, unsigned int unPort);

## 1.9 网络数据包

文档名：pluto.h pluto.cpp

模块名：pluto数据模块

类：CPluto

方法：

***Public：***

描述： 填充msgid到pluto，不做加密处理，处理Pluto头偏移量

CPluto& Encode(pluto\_msgid\_t msgid);

描述：重载<<符号，处理不同的数据类型

CPluto& operator<< (bool n);

CPluto& operator<< (uint8\_t n);

CPluto& operator<< (uint16\_t n);

CPluto& operator<< (uint32\_t n);

CPluto& operator<< (uint64\_t n);

CPluto& operator<< (int8\_t n);

CPluto& operator<< (int16\_t n);

CPluto& operator<< (int32\_t n);

CPluto& operator<< (int64\_t n);

CPluto& operator<< (float32\_t f);

CPluto& operator<< (float64\_t f);

CPluto& operator<< (const char\* s);

CPluto& operator<< (const string& s);

CPluto& operator<< (const charArrayDummy& r);

typedef CPluto& (\*pluto\_op) (CPluto&);

CPluto& operator<< (pluto\_op op);

friend CPluto& EndPluto(CPluto& p);

CPluto& operator<< (const CEntityMailbox& m);

CPluto& operator<< (const CEntityParent& e);

描述：用输入对象填充pluto

CPluto& FillPluto(const VOBJECT& v);

描述：用字符串填充pluto

CPluto& FillPlutoFromStr(VTYPE vt, const char\* s, unsigned long n);

描述：pluto数据解密操作，对msgid有一定的要求，大于临界值

CPluto& Decode();

描述：以下为pluto数据转储重载函数

CPluto& operator>> (bool& n);

CPluto& operator>> (uint8\_t& n);

CPluto& operator>> (uint16\_t& n);

CPluto& operator>> (uint32\_t& n);

CPluto& operator>> (uint64\_t& n);

CPluto& operator>> (int8\_t& n);

CPluto& operator>> (int16\_t& n);

CPluto& operator>> (int32\_t& n);

CPluto& operator>> (int64\_t& n);

CPluto& operator>> (float32\_t& f);

CPluto& operator>> (float64\_t& f);

CPluto& operator>> (charArrayDummy& d);

CPluto& operator>> (string& s);

CPluto& operator>> (CEntityMailbox& m);

描述：用pluto数据内容填充VOBJECT

void FillVObject(VTYPE vt, VOBJECT& v);

描述：解析pluto数据到实体属性中

bool UnpickleEntity(VOBJECT& v);

## 1.10 rpc处理

文档名：rpc\_mogo.h rpc\_mogo.cpp

模块名：rpc处理模块

类：CRpcUtil

方法：

***Public：***

描述：方法重载 rpc数据处理，最多可以添加六个typename参数

template<typename T1>

void Encode(CPluto& u, pluto\_msgid\_t msg\_id, const T1& p1);

template<typename T1, typename T2>

void Encode(CPluto& u, pluto\_msgid\_t msg\_id, const T1& p1, const T2& p2);

template<typename T1, typename T2, typename T3>

void Encode(CPluto& u, pluto\_msgid\_t msg\_id, const T1& p1, const T2& p2, const T3& p3);

template<typename T1, typename T2, typename T3, typename T4>

void Encode(CPluto& u, pluto\_msgid\_t msg\_id, const T1& p1, const T2& p2, const T3& p3, const T4& p4);

template<typename T1, typename T2, typename T3, typename T4, typename T5>

void Encode(CPluto& u, pluto\_msgid\_t msg\_id, const T1& p1, const T2& p2, const T3& p3, const T4& p4, const T5& p5);

template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6>

void Encode(CPluto& u, pluto\_msgid\_t msg\_id, const T1& p1, const T2& p2, const T3& p3, const T4& p4, const T5& p5, const T6& p6);

**注意：**本模块中定义了MSGID\_ENUM\_TYPE

## 1.11 定时器

文档名：timer.h timer.cpp

模块名：定时器模块

类：TimerData CTimerHeap

方法：

***Public：***

描述：根据参数生成一个timer data 并压入队列，返回timer id

int CTimerHeap::AddTimer(unsigned int nStart, int nInterval, TENTITYID nId, int nUserData)

## 1.12 类型定义模块

文档名：type\_mogo.h type\_mogo.cpp

模块名：类型定义模块

类： CEntityMailbox CClientMailbox VOBJECT

方法：

***Extern methods：***

描述：用0初始化VOBJECT的属性vv，前两个参数暂时没有用到

void FillVObjectInitValue(const CEntityParent\* p, const string& strAttri, VOBJECT& v);

描述： 用默认值初始化VOBJECT的属性vv，默认值为空则调用上个初始化函数

void FillVObjectDefaultValue(const CEntityParent\* p, const string& strAttri, VOBJECT& v, const string& strDefault);

描述：将参数s和v中的值以传入argv中idx索引项保存，v和s的长度保存于argvlen

void PushVObjectToRedisCmd(int idx, const string& s, VOBJECT& v, char\*\* argv, size\_t\* argvlen);

**注意：**本文件中定义了VVALUE、VTYPE

## 1.13 工具包

文档名：util.h util.cpp

模块名：工具封装模块

类： CGetTimeOfDay

方法：

***Public：***

描述：获取当前时间与上个时间点的差

int GetLapsedTime()

***Extern mehods：***

**字符串操作方法：**

描述:除去字符串左边的空格，参数为string型

string& Ltrim(string& s);

描述：除去字符串左边的空格，参数为char\*型

char\* Ltrim(char\* p);

描述：除去字符串右边的空格，参数为string型

string& Rtrim(string& s);

描述：除去字符串左边的空格，参数为char\*型

char\* Rtrim(char\* p);

描述：除去字符串左右两边的空格，参数为string型

string& Trim(string& s)

描述：除去字符串左右两边的字符串，参数为char\*型

char\* Trim(char\* s)

描述：比较两个字符串大写是否匹配

bool UpperStrCmp(const char\* src, const char\* desc);

描述：将字符串以隔符nDelim拆分为子字符串并放入队列中返回

list<string> SplitString(const string& s1, int nDelim);

描述：将字符串以隔符nDelim拆分为子字符串并放入指定队列中

void SplitString(const string& s1, int nDelim, list<string>& ls);

描述：将字符串s中第一次出现的子串pszSrc替换为pszRep

string& xReplace(string& s1, const char\* pszSrc, const char\* pszRep);

描述：判断字符串pszStr是否全部由数字字符组成

bool IsDigitStr(const char\* pszStr);

**文件操作：**

描述：判断文件pszFileName是否存在

bool IsFileExist(const char\* pszFileName);

描述：判断文件pszFileNames是否存在

bool IsFileExist(const string& strFileName);

描述：检查目录是否存在，不存在则创建

void CheckDir(const char\* pszDirName, bool bLog = false);

描述：判断一个配置文件读取的路径名最后是否带路径分隔符，如果没有则加上

string FormatPathName(const string& strPath);

**容器操作：**

描述：清楚容器内容

template <typename TP, template <typename ELEM,

typename ALLOC = std::allocator<ELEM> > class TC>

void ClearContainer(TC<TP, std::allocator<TP> >& c1)

描述：清楚映射内容

template<typename T1, typename T2,

template <class \_Kty, class \_Ty, class \_Pr = std::less<\_Kty>,

class \_Alloc = std::allocator<std::pair<const \_Kty, \_Ty> > > class M >

void ClearMap(M<T1, T2, std::less<T1>, std::allocator<std::pair<const T1, T2> > >& c1)

**时间操作：**

描述：获取当前时间

void GetCurTime(string& strCurTime);

描述：获取明日此时的时间

string GetNextTime(const string& strLastTime);

描述：获取昨日的此时的时间

void GetYesterday(string& strYesterday);

描述：判断是否为同一天

bool DayDiff(const string& strDayTime, int nClock);

描述：获取当前的日期时间

void GetDateTime(char\* pszDT, size\_t nLen);

## 1.14 world模块

文档名：world.h world.cpp

模块名：

类： CMailBoxManager

方法：

***Public：***

描述：返回mailbox队列

list<CMailBox\*>& GetMailboxs()；

类：world

方法：

Public：

描述：重载rpccall 方法，由CRPCUtil预处理（包括加密环节），然后由nserverId取得服务器mailbox，调用mailbox 的rpccall继续处理。

template<typename T1>

void RpcCall(uint16\_t nServerId, pluto\_msgid\_t msg\_id, const T1& p1);

template<typename T1, typename T2>

void RpcCall(uint16\_t nServerId, pluto\_msgid\_t msg\_id, const T1& p1, const T2& p2);

template<typename T1, typename T2, typename T3>

void RpcCall(uint16\_t nServerId, pluto\_msgid\_t msg\_id, const T1& p1, const T2& p2, const T3& p3);

template<typename T1, typename T2, typename T3, typename T4>

void RpcCall(uint16\_t nServerId, pluto\_msgid\_t msg\_id, const T1& p1, const T2& p2, const T3& p3, const T4& p4);

template<typename T1, typename T2, typename T3, typename T4, typename T5>

void RpcCall(uint16\_t nServerId, pluto\_msgid\_t msg\_id, const T1& p1, const T2& p2, const T3& p3, const T4& p4, const T5& p5);

template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6>

void RpcCall(uint16\_t nServerId, pluto\_msgid\_t msg\_id, const T1& p1, const T2& p2, const T3& p3, const T4& p4, const T5& p5, const T6& p6);

描述：将pluto放入邮箱发送队列，如果nServerId是服务器自身的，直接加入到接收队列。

bool PushPlutoToMailbox(uint16\_t nServerId, CPluto\* u);

描述：检查client的RPC调用是否合法

bool CheckClientRpc(CPluto& u);

描述：将客户端的socket fd附加到pluto解包出来的数据结构上

void AddClientFdToVObjectList(int fd, T\_VECTOR\_OBJECT\* p);

描述：检查client是否为可信的

bool IsTrustedClient(const string& strClientAddr);

描述：将实体tid添加到时间保存队列

void RegisterTimeSave(TENTITYID tid);

## 1.15 world login模块

文档名：world\_login.h world\_login.cpp

模块名：

类： CWorldLogin inheritance world

方法：

***Public：***

描述：调用当前对象的方法实现，继承于父类world的虚方法。首先是从pluto中获取message id，然后检查message id是否合法，由CRPCUtil对Pluto解码出T\_VECTOR\_OBJECT结构，找到调用对应的方法并将解码的参数传入，执行完后返回。

int FromRpcCall(CPluto& u);

***Private：***

描述：处理登陆服务，首先检查无服务是否可用，不可用返回给请求端不可用数据；可用则将连接的fd和account保存，传递客服端信息给DBMGR。

注：为前期服务器与客服端之间的连接，在此修改了逻辑直接将数据回传给客服端。

int AccountLogin(T\_VECTOR\_OBJECT\* p);

描述：处理DBMGR登陆回调，响应给客服端登陆状态。

int SelectAccountCallback(T\_VECTOR\_OBJECT\* p);

描述：通知客服端连接服务器BaseApp

int NotifyClientToAttach(T\_VECTOR\_OBJECT\* p);

描述：修改服务器登陆服务标志，允许客服端连接服务

int ModifyLoginFlag(T\_VECTOR\_OBJECT\* p);

## 1.16 global world

文档名：world\_select.h

模块名：全局world

类： 无

方法：

***extern methods：***

描述：返回全局world对象

world\* GetWorld()

# 第二章 模块执行流程

本章主要介绍testserver这个测试服务环境从启动到RPC调用再到服务器关闭的系列过程的详细讲解。帮助大家尽快熟悉框架的结构和程序执行的具体流程。

程序从main函数开始，在此之前实例化了一个全局worldlogin对象，供执行期间调用。worldlogin对象的属性主要用来维护的是客服端连接和账号之间的映射关系。

world\* g\_pTheWorld = new CWorldLogin();

## 2.1配置文件

启动服务器的时候需要传递三个参数：分别是配置文件，当前服务器对应的id，日志文件存放路径（例如，可执行文件为main，配置文件为./cfg.ini 当前服务器id分配为1， 日志文件存放路径为./log/， 则有： root@usr ~# ./main ./cfg.ini 1 ./log/）。

配置文件格式如下：

[server\_all]

server\_count = n

[server\_i]( 1 <= i <= n)

type = value

ip = value

port = value

//以下为绿色通道配置

[clients]

trusted = 192.168.1.1, 192.168.1.2, ...

...

//以下为xml文件配置路径

[init]

def\_path=~/entity\_defs

log\_path=./log

服务器类型分为六种：loginapp =1 cwmd=2 dbmgr=3 timerd=4 baseapp=6 cellapp=7

编号为5的分给客服端， baseapp和cellapp都可以有多个进程存在，其他的进程只能存在一个。

配置文件初始化只要完成建立与各个服务器之间建立邮件通信基础。

框架编译相关库：tinyxml、 lua、redis、openssl等

## 2.2 运行环境初始化

以下根据实际语句解析服务执行流程：

①g\_logger.SetLogPath(pszLogPath);

设置日志文件存放路径

②world& the\_world = \*GetWorld();

获取全局world对象，对每个服务进程都有一个独立的全局world对象，用这个对象来 管理与服务器的所有连接。

③the\_world.init(pszEtcFn);

完成配置文件的初始化，并与配置文件中所存在服务器建立对应的mailbox，放入邮箱 管理对象中。解析实体对象（entities.xml）和实体定义（\*.def）,其中\*号代表实体名称从entities.xml中取得。

④CLoginappServer s;

实例化一个CLoginappServer对象

⑤s.SetMailboxId(nServerId);

用传入的服务器编号初始化CLoginappServer对象自己的邮箱编号

⑥s.SetWorld(&the\_world);

⑦the\_world.SetServer(&s);

以上两步主要完成world与server之间的映射关系

⑧uint16\_t unPort = the\_world.GetServerPort(nServerId);

根据服务器编号获取服务器的连接端口号。

⑨s.Service("", unPort);

传入端口号启动server的service，自此真正开始建立服务，这里传入地址为空是让系统自动选择。

**注意：**这些内容都包含在main.cpp文件中，用来完成系统的初始化工作。非常重要的一部分是在world实例化的时候会完成所有远程方法调用的初始化（两部分：参数的格式和方法id，参数值在接收的pluto 中解析获得），这一部分内容包含在rpc\_mogo.cpp文件中。

## 2.3 服务启动流程

由2.2节中第九步开始进入网络服务器建立和数据收发处理。

在server中由四步组成：

①Start Server

启动服务主要完成Epoll池的建立。

②Connect Mailboxs

这一步主要完成配置文件中解析出来的各服务器之间的连接，数据通信由mailbox管理， mailbox由manager mailbox 管理。

③循环处理数据

完成新链接事件的处理，添加新的链接到队列中并注册到Epoll池中；接受网络数据转换为pluto添加到接受队列；发送mailbox中的pluto数据队列；处理重连接队列；处理超时服务。在链接处理是的时候有三个不同类型：

FD\_TYPE\_SERVER = 1 本服务器的当前进程类型，在StartServer中设置

FD\_TYPE\_MAILBOX = 2 指主动向其他服务器或需要通信进程发起链接的类型

FD\_TYPE\_ACCEPT = 3 其他服务向本进程发起链接的类型

服务流程是其他进程或客服端向本进程发起连接由本进程监听，设置连接为FD\_TYPE\_ACCEPT，数据接收就转到FD\_TYPE\_ACCEPT处理；若是类型为FD\_TYPE\_MAILBOX直接接收数据。

④shutdown server

关闭服务器的相关处理。

## 2.4 rpc处理

rpc处理的基本工作是由handlepluto开始的，所有收发的数据都是一个pluto对象，进程中的接收的数据都保存在本进程的接收队列，处理的方式按照队列的先后顺序处理。

从队列中弹出一个pluto进入FromRpcCall中处理，FromRpcCall是worldlogin继承与world的纯虚函数，对每个不同进程都有自己的不同实现（这里关注的本进程的实现）。

pluto处理基本流程如下：

①从pluto中解码出pluto\_msg\_id，解码这个id的时候偏移量是不做移动的，用固定的消息头得到的。

②调用CheckClientRpc校验pluto\_msg\_id是否合法。

③由rpc对象对pluto对象的Decode返回解码方法参数对象的队列。

**特别注意**：并不是所有的消息都需要加密和解密，有一个设定的值定义为MAX\_CLIENT\_SERVER\_MSGID，低于这值所有消息都需要做加密处理（解码和加密是两回事，不要混淆）。

进入rpc.decode后还要进入pluto.decode,里面很重要的一个细节是要执行以下语句：

m\_unLen = MSGLEN\_HEAD + MSGLEN\_RESERVED + MSGLEN\_MSGID;

执行这条语句后，才能取得真正的参数值开始位置。不然解码的时候永远是错的，因为m\_unLen会一直保持为pluto的总长度。

方法参数队列的构建流程：

首先在rpc的msg\_id与实体方法的映射map<pluto\_msgid\_t, \_SEntityDefMethods\*>中查询是否存在msg\_id这个映射， 检查是否为来自客服端对base或cell的实体的方法调用。两种过程大同小异，这里只做msg\_id存在的分解处理。

从映射中取出SEntityDefMethods，对他的每一个参数类型从pluto中解出来，构成一个VOBJECT插入队列中。

④根据msg\_id进入对应的处理方法。

⑤数据善后处理。