语言: C++/LUA

数据库: SSDB+Mysql

开发测试平台:windows

发布平台:linux

服务器配置: ServerConf.lua

1. 协议目录: Data/Protobuf

通常1个功能定义1个proto文件。

生成协议工具: 协议.bat。

流程:

1. 写好proto文件。

2. 执行 协议.bat，服务器会生成到LoadPBCProto.lua中，

客户端会生成到客户端对应目录。

3. 服务器在CmdDef.lua里面注册通讯RPC, 并拷贝到Data/Document/协议文档.xls。

4. 客户端从协议文档.xls中拷贝通讯RPC到cmdDef.js里面。

5. 请求协议格式: xxxReq，返回协议格式:xxxRet。

2. 配置文件目录: Data/Config

CSV 需要C++读取的配XML表。

XML 生产LUA/JS文件用的XML配表。

工具: 导表.bat。

流程:

1. 服务器定义好表格式，填临时数据。

2. 执行 导表.bat，会把生成的LUA文件拷贝到服务器对应目录，

JS文件拷贝到客户端对应目录。

3. 客户端如果需要添加配表字段，就修改xml文件，再跑流程。

3. GM指令文件：Data/Document/GM.txt

目的: 方便测试用

4. 服务器客户端通讯约定:

通讯类文件约定: (文件名)Rpc.lua

发送消息给客户端例子: CmdNet.PBSrv2Clt(协议名, 目标服务器，目标会话ID, 消息体)

5. 服务器注册/取消计时器:

local nTickID = goTimerMgr:Interval(秒, functin() 业务 end)

goTimerMgr:Clear(nTickID)

注意：不需要了的计时器一定要调用Clear释放，否则会泄露。

1. 服务器间通讯组件: goRemoteCall

goRemoteCall:Call(sCallFunc, nTarServer, nTarService, nTarSession, ...) 异步调用远程方法无返回回调；

goRemoteCall:CallWait(sCallFunc, fnCallback, nTarServer, nTarService, nTarSession, ...) 异步调用远程方法，有返回回调。

知识点:

1. Google Protobuffer （通讯协议）

2. Common/LuaClass.lua （模拟面向对象）

服务器组成：

因为需要支持跨服功能，所以分三个组服务器:

\_RoutServer 服务器内部路由(包括本地服和世界服之间)

\_WorldServer 世界服(WGlobalServer, LogicServer)

\_LocalServer 本地服(GlobalServer, LogServer, LoginServer,LogicServer,GateServer)

路由服务 RouterServer.exe 服务器内部路由

网关服务 GateServer.exe 玩家链接处理

登陆服武 LoginServer.exe 处理玩家登陆

逻辑服务 LogicServer.exe 场景,战斗,玩家身上模块等(注意也有跨服场景服)

全局服务 GlobalServer.exe 本地全局数据,如活动,联盟等

世界全局服务 WGlobalServer.exe 跨服全服数据,如跨服组队,跨服聊天等

日志服务 LogServer.exe 写运营日志到Mysql

世界服和本地服，每个服务进程都有唯一的服务ID

LogicServer目录结构:

ConfCheck：

启动的时候需要对配置文件进行校验时写在这里。

Dup: 场景,副本模块。

Monster: 怪物模块。

Module：

玩家身上的系统(背包，宠物，技能，装备等)。

Player：

玩家模块，所有的Module对象会挂在玩家对象身上。玩家上线的时候会创建玩家对象和Module对象，并加载自己和所有Module对象的数据。离线时会保存自己和Moudle的脏数据，一定时间后会将玩家对象和Module对象清楚。

Common目录(重点看)：

定义了一些公共的函数，组件，算法等

DB目录(SSDB工作目录):

游戏数据存放的地方

批处理相关：

提交SVN

提交.bat

启动服务器批处理：

启动.bat

关闭服务器批处理:

关闭.bat

生成协议批处理:

协议.bat

导表批处理:

导表.bat

LuaTrace 和 Print区别:

LuaTrace: 无论是debug或者release都会打印

Print: debug会打印 release不会打印

平时调试(debug)不重要的打印用Print，如果上线后(release)也需要打印的用LuaTrace

目的就是上线后屏蔽掉不必要的打印，打印耗性能

代码风格：前缀用小写，其他的每个单词首字母大写

变量前缀: n(number), s(string), t(table),fn(function),o(object)

全局前缀: g

类: CXxx