# 易栈客户端

## 基础知识

### 网络数据交互格式

随着网络的不断扩张，网络数据变得纷繁复杂，网络数据交互的复杂性也越来越严重。

为了快速而简洁的进行网络数据交互，各种各样的网络数据格式应运而生。

从html到xml，从xml到json，不断推陈出新。

也许走上工作岗位之后，各位还会在各行各业接触到行业内的各种数据格式，但是下面的两种数据格式，是我要给大家讲解和使用的。

希望大家能够明白，**如何快速学习一个新的数据格式**。

#### Json（基于JsonCpp）

格式全称：JavaScript Object Notation

格式特点：

1. **与开发语言无关**
2. 轻量级的数据存储格式
3. 易于解析和封装
4. 数据对象内部一般采用key-value模式来存储
5. 支持数组、字符串、整数、小数、对象、布尔、空
6. 可以用于数据传输、*数据保存*、数据交换

格式详解：

{

"name":"张三",

"age":18,

"身高":1.81,

"生日":"2002-01-02",

"学校":"北大青鸟",

"专业":["厨师","挖掘机"],

"单身":true,

"地址":null,

“好友”:{...}

}

开源项目：

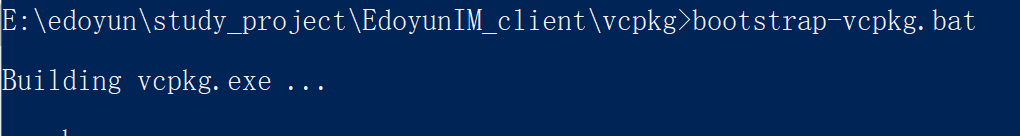
通用版本：（推荐）

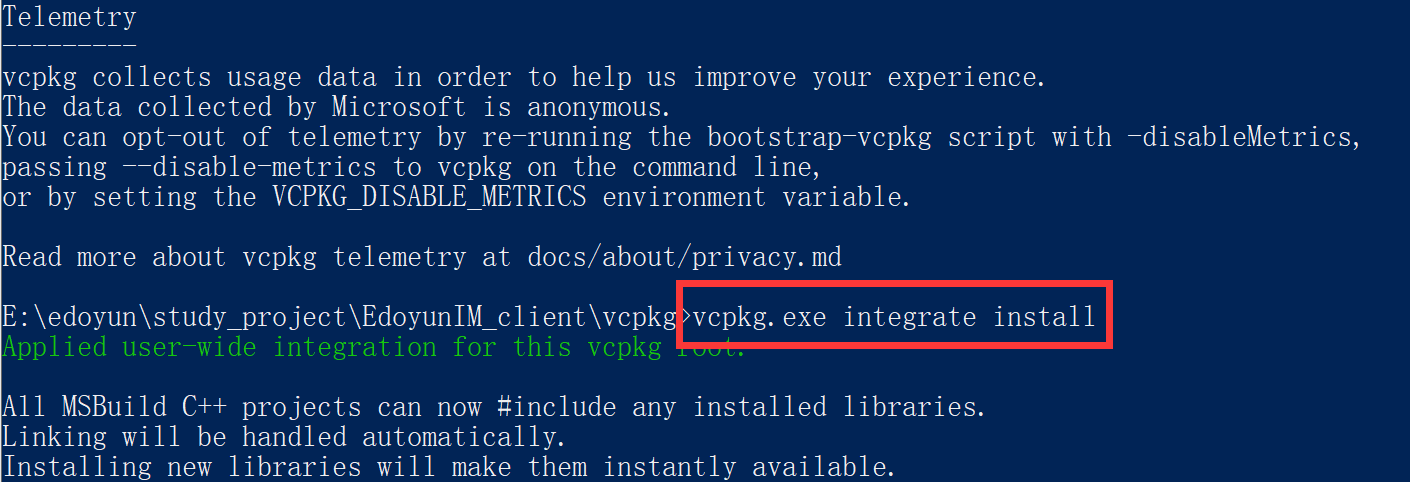
<https://github.com/open-source-parsers/jsoncpp>

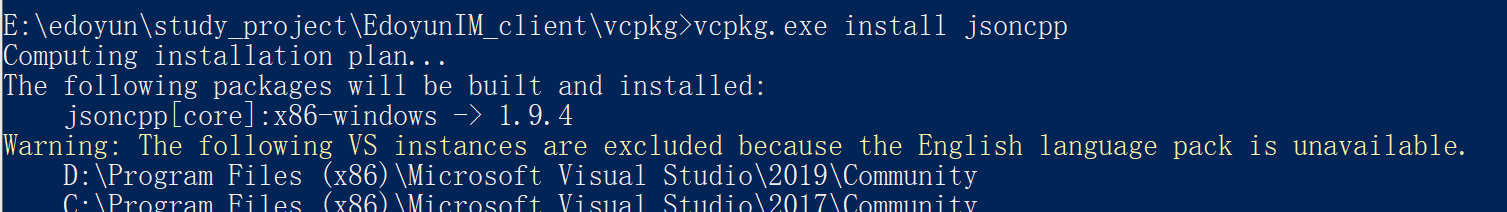
Window专用版本：

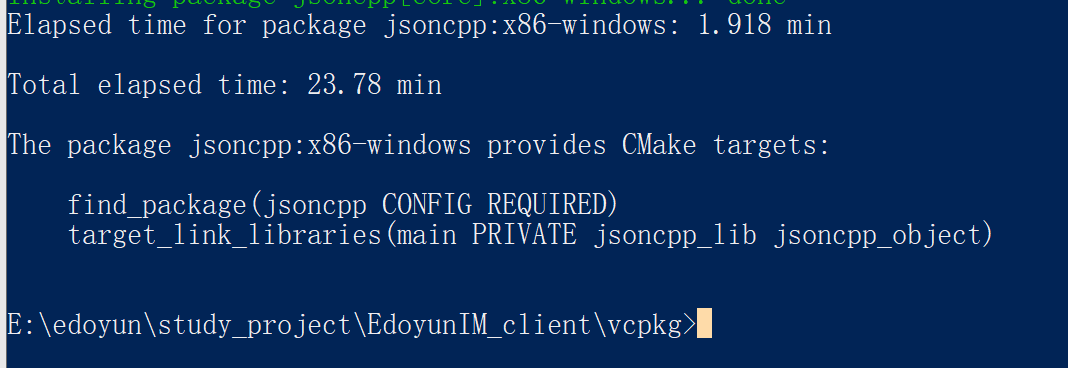
<https://github.com/Microsoft/vcpkg.git>

<https://gitee.com/yao_yu_qq/vcpkg>









上面的版本使用都有一些麻烦

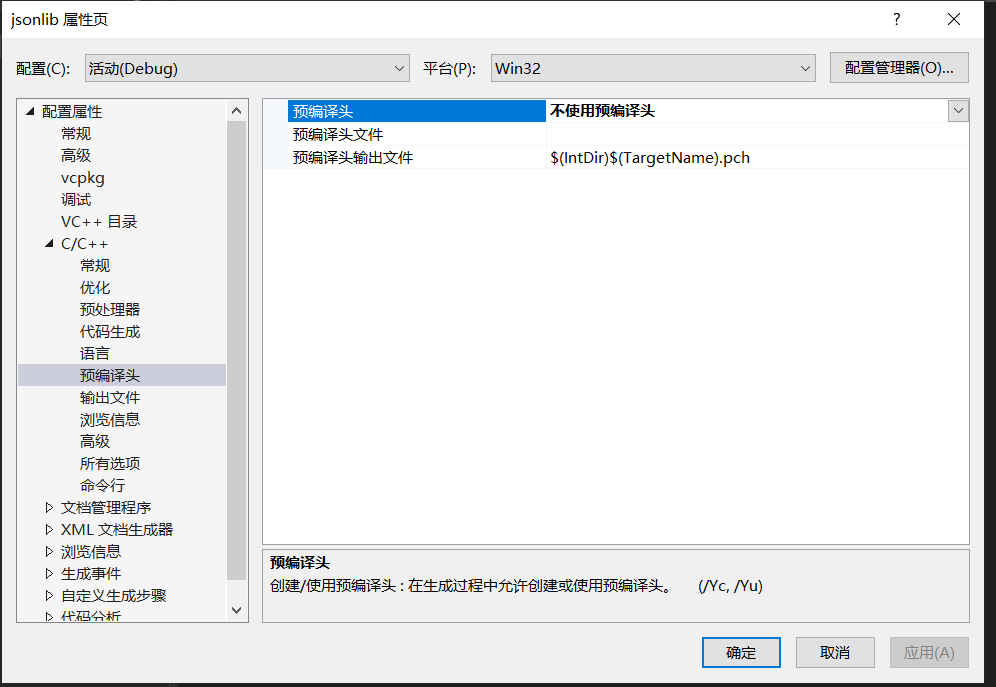
**最好使用易道云学院整理版本**

可以做成库集成，也可以直接吸纳源代码

使用说明：

项目编译（编译成静态库）

不使用预编译头



文件配置

版本说明

项目集成

引用说明

使用说明

1 包含头文件

2 引用库文件

3 使用Reader、Value类

参考示例代码

#### Xml（基于.COM）

##### COM的介绍

组件化解决的问题是拿来就用，不需要太多的额外编译，就能够进行适配

仅仅引用头文件，加上一个可运行的程序，就可以到处使用，随时引用

甚至不要头文件，直接拿来用

COM里面有接口规范，可以通过接口查询方法

本地COM服务（微服务器），COM控件需要先注册后使用（dll库、ocx库）

ActiveX把COM的升级为DCOM

MSXML对应的头文件<msxml2.h>

##### IXMLDOMNode

get\_nodeName

get\_nodeValue

put\_nodeValue

get\_nodeType

get\_parentNode

get\_childNodes

get\_firstChild

get\_lastChild

get\_previousSibling

get\_nextSibling

get\_attributes

insertBefore

replaceChild

removeChild

appendChild

hasChildNodes

get\_ownerDocument

cloneNode

get\_nodeTypeString

get\_text

put\_text

get\_specified

get\_definition

get\_nodeTypedValue

put\_nodeTypedValue

get\_dataType

put\_dataType

get\_xml

transformNode

selectNodes

selectSingleNode

get\_parsed

get\_namespaceURI

get\_prefix

get\_baseName

transformNodeToObject

##### IXMLDOMNodeList

get\_item

get\_length

nextNode

reset

get\_\_newEnum

##### IXMLDOMDocument2

get\_namespaces

get\_schemas

putref\_schemas

validate

setProperty

getProperty

### UI库

WTL的必要性

MFC 兼容性导致其性能下降、影响架构设计

有一定的封装基础、不是mfc的框架、能够自己定制一些窗口行为

WTL对模板编程有一点要求

#### Wtl

##### ATL GUI类

ATL Activity Template Library .NET

##### WTL GUI基类

WTL Windows Template Library

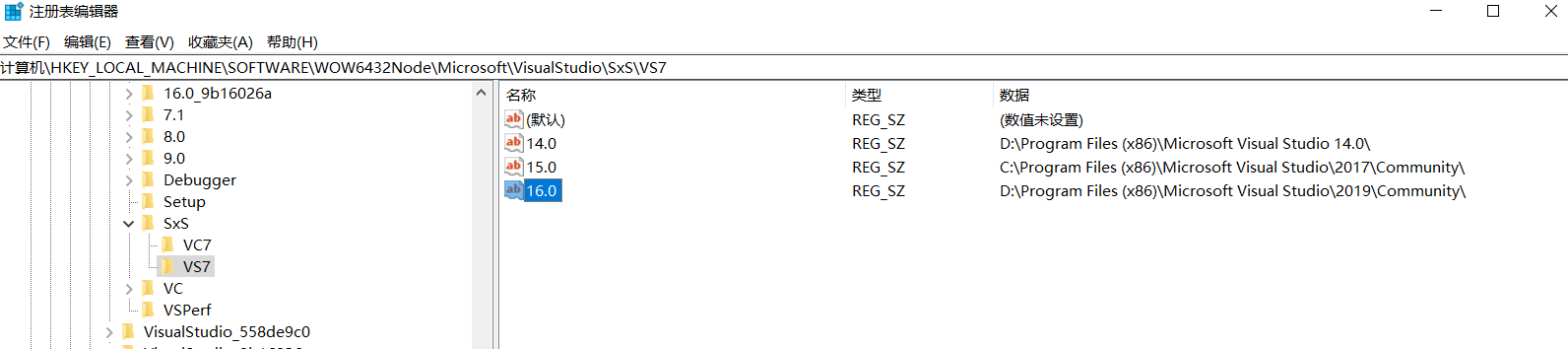
WTL类的类型：

1. 主框架窗口 CFrameWindowImpl CMDIFrameWindowImpl
2. 控件 CButton,CListViewCtrl等等
3. GDI对象 CDC、CMenu等等
4. 特殊界面特性 CSplitterWindow, CUpdateUI, CDialogResize, CCustomDraw
5. 实用的工具类和宏 CString, CRect, BEGIN\_MSG\_MAP\_EX 等等

##### 框架类

WTL下载地址：

<https://sourceforge.net/projects/wtl/files/latest/download>



// Search for Visual Studio 2005-2015

var nVersions = 7;

var astrRegKeyVer = new Array();

astrRegKeyVer[0] = "Microsoft\\VisualStudio\\8.0\\Setup\\VC\\ProductDir";

astrRegKeyVer[1] = "Microsoft\\VisualStudio\\9.0\\Setup\\VC\\ProductDir";

astrRegKeyVer[2] = "Microsoft\\VisualStudio\\10.0\\Setup\\VC\\ProductDir";

astrRegKeyVer[3] = "Microsoft\\VisualStudio\\11.0\\Setup\\VC\\ProductDir";

astrRegKeyVer[4] = "Microsoft\\VisualStudio\\12.0\\Setup\\VC\\ProductDir";

astrRegKeyVer[5] = "Microsoft\\VisualStudio\\14.0\\Setup\\VC\\ProductDir";

astrRegKeyVer[6] = "Microsoft\\VisualStudio\\SxS\\VS7\\16.0";

var strWizardFolder = "vcprojects";

var astrVersions = new Array();

astrVersions[0] = "Visual Studio 2005 (8.0)";

astrVersions[1] = "Visual Studio 2008 (9.0)";

astrVersions[2] = "Visual Studio 2010 (10.0)";

astrVersions[3] = "Visual Studio 2012 (11.0)";

astrVersions[4] = "Visual Studio 2013 (12.0)";

astrVersions[5] = "Visual Studio 2015 (14.0)";

astrVersions[6] = "Visual Studio 2019 (16.0)";

var astrWizVer = new Array();

astrWizVer[0] = "8.0";

astrWizVer[1] = "9.0";

astrWizVer[2] = "10.0";

astrWizVer[3] = "11.0";

astrWizVer[4] = "12.0";

astrWizVer[5] = "14.0";

astrWizVer[5] = "16.0";

var astrParamVer = new Array();

astrParamVer[2] = "10";

astrParamVer[3] = "11";

astrParamVer[4] = "12";

astrParamVer[5] = "14";

astrParamVer[5] = "16";

##### 工具栏和状态栏

##### 对话框和控件

四种模式：

控件类Attach模式

CContainedWindow模式

子类模式

DDX模式

#### SkinLib

1. MESSAGE\_RANGE\_HANDLER\_EX 响应范围内的消息
2. 一个消息，被多次注册，则会多次响应
3. 控件往往都是一些贴图、一些光影游戏
4. 控件的外形，也可以进行修改，可以进行组合（与、或、非、异或）

##### CSkinEdit

##### CSkinButton

##### CSkinStatic

##### CSkinComboBox

##### CSkinComboBox\_Edit

##### CSkinComboBox\_ListBox

##### CSkinComboBox\_Static

##### CSkinDialog

##### CSkinListCtrl

##### CSkinHyperLink

##### CSkinManager

##### CSkinMenu

##### CSkinMessageBox

##### CSkinPictureBx

##### CSkinRichEdit

##### CSkinScrollBar

### 数据库

#### Sqlite3

##### 特点

1. 体积小
2. 支持window和Linux（包括Android），跨平台性很好
3. 采用的一种库函数的形式来提供数据库的服务
4. 文件性的数据库
5. 缺点，不能使用网络，单纯就是本地数据库
6. 加密性能不是很好，安全性不足

##### Sql语法

插入

INSERT INTO 表名 (列名1,...)

VALUES (列1值,...);

修改

UPDATE 表名

SET 列名1= 列1值, ....

WHERE [条件表达式];

删除

DELETE FROM 表名

WHERE [条件表达式];

查询

SELECT 列名1, .... FROM 表名;

WHERE [条件表达式];

` “ ‘

创建表

CREATE TABLE 库名.表名(

列1名 类型 特征（主键、唯一、非空、自增、注释）,

...

);

删除表

DROP TABLE 表名;

##### 接口函数

如果文件存在，则尝试打开

如果文件不存在，则创建

这个文件就是一个库

不支持多线程访问，不支持多实例访问

SQLITE\_API int sqlite3\_open(

const char\*filename, /\* Database filename (UTF-8)\*/

sqlite3\*\*ppDb /\* OUT: SQLite db handle\*/

);

SQLITE\_API int sqlite3\_open16(

const void\*filename, /\* Database filename (UTF-16)\*/

sqlite3\*\*ppDb /\* OUT: SQLite db handle\*/

);

SQLITE\_API int sqlite3\_close(sqlite3\*);

SQLITE\_API int sqlite3\_exec(

sqlite3\*, /\* An open database\*/

const char\*sql, /\* SQL to be evaluated\*/

int (\*callback)(void\*,int,char\*\*,char\*\*), /\* Callback function\*/

void\*, /\* 1st argument to callback\*/

char\*\*errmsg /\* Error msg written here\*/

);

SQLITE\_API int sqlite3\_prepare(

sqlite3\*db, /\* Database handle\*/

const char\*zSql, /\* SQL statement, UTF-8 encoded\*/

int nByte, /\* Maximum length of zSql in bytes.\*/

sqlite3\_stmt\*\*ppStmt, /\* OUT: Statement handle\*/

const char\*\*pzTail /\* OUT: Pointer to unused portion of zSql\*/

);

SQLITE\_API int sqlite3\_bind\_blob(sqlite3\_stmt\*, int, const void\*, int n, void(\*)(void\*));

SQLITE\_API int sqlite3\_bind\_double(sqlite3\_stmt\*, int, double);

SQLITE\_API int sqlite3\_bind\_int(sqlite3\_stmt\*, int, int);

SQLITE\_API int sqlite3\_bind\_int64(sqlite3\_stmt\*, int, sqlite3\_int64);

SQLITE\_API int sqlite3\_bind\_null(sqlite3\_stmt\*, int);

SQLITE\_API int sqlite3\_bind\_text(sqlite3\_stmt\*, int, const char\*, int n, void(\*)(void\*));

SQLITE\_API int sqlite3\_bind\_text16(sqlite3\_stmt\*, int, const void\*, int, void(\*)(void\*));

SQLITE\_API int sqlite3\_bind\_value(sqlite3\_stmt\*, int, const sqlite3\_value\*);

SQLITE\_API int sqlite3\_bind\_zeroblob(sqlite3\_stmt\*, int, int n);

SQLITE\_API int sqlite3\_step(sqlite3\_stmt\*);

SQLITE\_API int sqlite3\_finalize(sqlite3\_stmt\*pStmt);

SQLITE\_API const void\*sqlite3\_column\_blob(sqlite3\_stmt\*, int iCol);

SQLITE\_API int sqlite3\_column\_bytes(sqlite3\_stmt\*, int iCol);

SQLITE\_API int sqlite3\_column\_bytes16(sqlite3\_stmt\*, int iCol);

SQLITE\_API double sqlite3\_column\_double(sqlite3\_stmt\*, int iCol);

SQLITE\_API int sqlite3\_column\_int(sqlite3\_stmt\*, int iCol);

SQLITE\_API sqlite3\_int64 sqlite3\_column\_int64(sqlite3\_stmt\*, int iCol);

SQLITE\_API const unsigned char\*sqlite3\_column\_text(sqlite3\_stmt\*, int iCol);

SQLITE\_API const void\*sqlite3\_column\_text16(sqlite3\_stmt\*, int iCol);

SQLITE\_API int sqlite3\_column\_type(sqlite3\_stmt\*, int iCol);

SQLITE\_API sqlite3\_value\*sqlite3\_column\_value(sqlite3\_stmt\*, int iCol);

### 常用开源库

#### MD5

***本质上不属于加密算法***

不可逆

**但是它和加密紧密相连**

数字签名

唯一性（只有掌握密码的人才能签名出有效的结果）

不可更改性（一旦更改，签名将发生变化）

文件防止篡改

用户登录

安全性：一般认为是安全

攻击手段：撞库

针对简单密码的情况

12345678 建立一个库

不要用简单密码，不要用常见密码

MD5算法说明

/\* F, G, H and I are basic MD5 functions.\*/

#define F(x, y, z) (((x) & (y)) | ((~x) & (z)))

#define G(x, y, z) (((x) & (z)) | ((y) & (~z)))

#define H(x, y, z) ((x) ^ (y) ^ (z))

#define I(x, y, z) ((y) ^ ((x) | (~z)))

/\* FF, GG, HH, and II transformations for rounds 1, 2, 3, and 4.

Rotation is separate from addition to prevent recomputation.\*/

#define FF(a, b, c, d, x, s, ac) { \

(a) += F ((b), (c), (d)) + (x) + (UINT4)(ac); \

(a) = ROTATE\_LEFT ((a), (s)); \

(a) += (b); \

}

#define GG(a, b, c, d, x, s, ac) { \

(a) += G ((b), (c), (d)) + (x) + (UINT4)(ac); \

(a) = ROTATE\_LEFT ((a), (s)); \

(a) += (b); \

}

#define HH(a, b, c, d, x, s, ac) { \

(a) += H ((b), (c), (d)) + (x) + (UINT4)(ac); \

(a) = ROTATE\_LEFT ((a), (s)); \

(a) += (b); \

}

#define II(a, b, c, d, x, s, ac) { \

(a) += I ((b), (c), (d)) + (x) + (UINT4)(ac); \

(a) = ROTATE\_LEFT ((a), (s)); \

(a) += (b); \

}

#### TuNzIp

Phil Katz（1962-2000.4）

Arc → pkarc

Zip



HZIP OpenZip(const TCHAR\* fn, const char\* password)

ZRESULT GetZipItem(HZIP hz, int index, ZIPENTRY\* ze);

ZRESULT FindZipItem(HZIP hz, const TCHAR\* name, bool ic, int\* index, ZIPENTRY\* ze);

ZRESULT UnzipItem(HZIP hz, int index, const TCHAR\* fn);

ZRESULT UnzipItem(HZIP hz, int index, void\* z, UINT len);

ZRESULT UnzipItemHandle(HZIP hz, int index, HANDLE h);

ZRESULT SetUnzipBaseDir(HZIP hz, const TCHAR\* dir);

ZRESULT CloseZip(HZIP hz);

UINT FormatZipMessage(ZRESULT code, TCHAR\* buf, UINT len);

#### Protobuf

Protobuf的下载与编译

<https://github.com/protocolbuffers/protobuf/tree/2.6.1-artifacts>

编译错误：

1 hash\_map需要宏 \_SILENCE\_STDEXT\_HASH\_DEPRECATION\_WARNINGS;

2 \_SILENCE\_TR1\_NAMESPACE\_DEPRECATION\_WARNING;

3 ETC宏 SPEPARATOR宏 注意加空格

4 修改编译，每次只编译1个项目

Protobuf的使用

数据封包，解包，打包是快捷，高效的

这个东西是比较容易修改的

与语言无关、平台无关、可扩展的

用于序列化结构化数据的

效率高

写一个协议文件

message EmbeddedMessage{

required uint32 nSubCmd = 1;

required string strParams = 2;

}

message EdyunMsg{

//required optional repeated

required uint32 nCmd = 1 [default = 0];

required string strData = 2;

optional EmbeddedMessage subMsg = 3;

}

输入命令行：

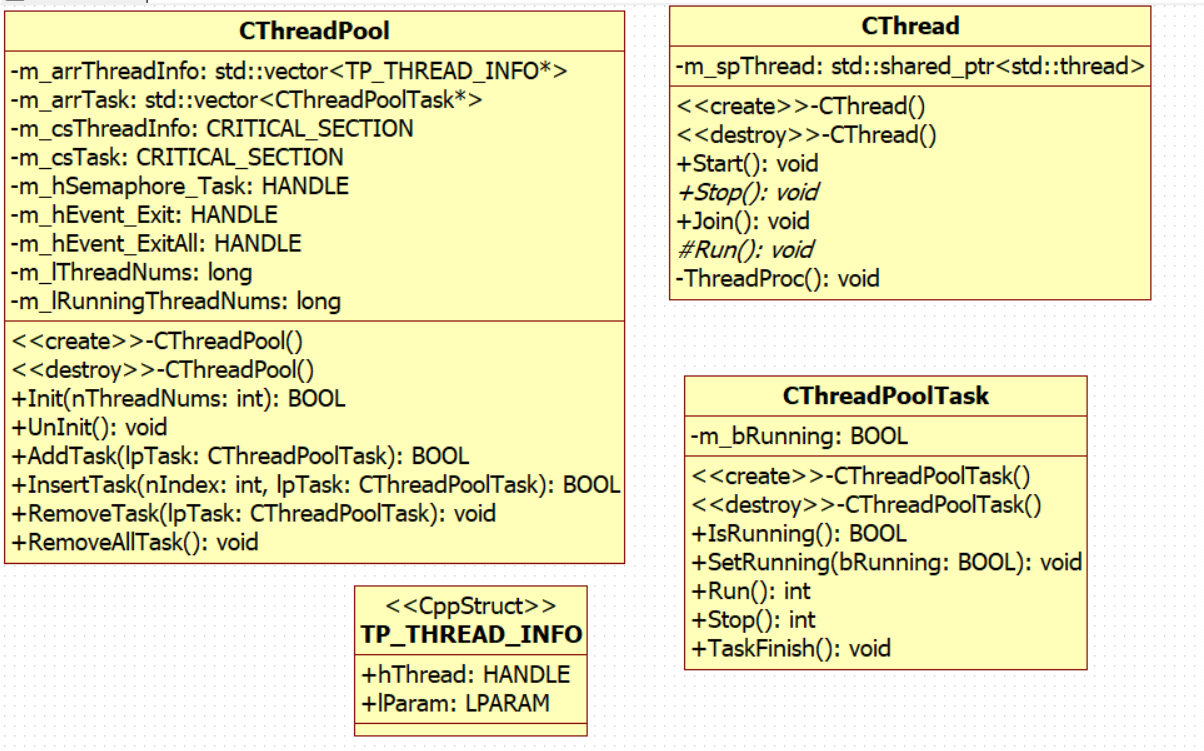
protoc.exe -I. --cpp\_out=. edoyun.proto

得到proto源码和头文件

## 项目设计与实现

### 基础模块设计与实现

#### 线程池的设计与实现

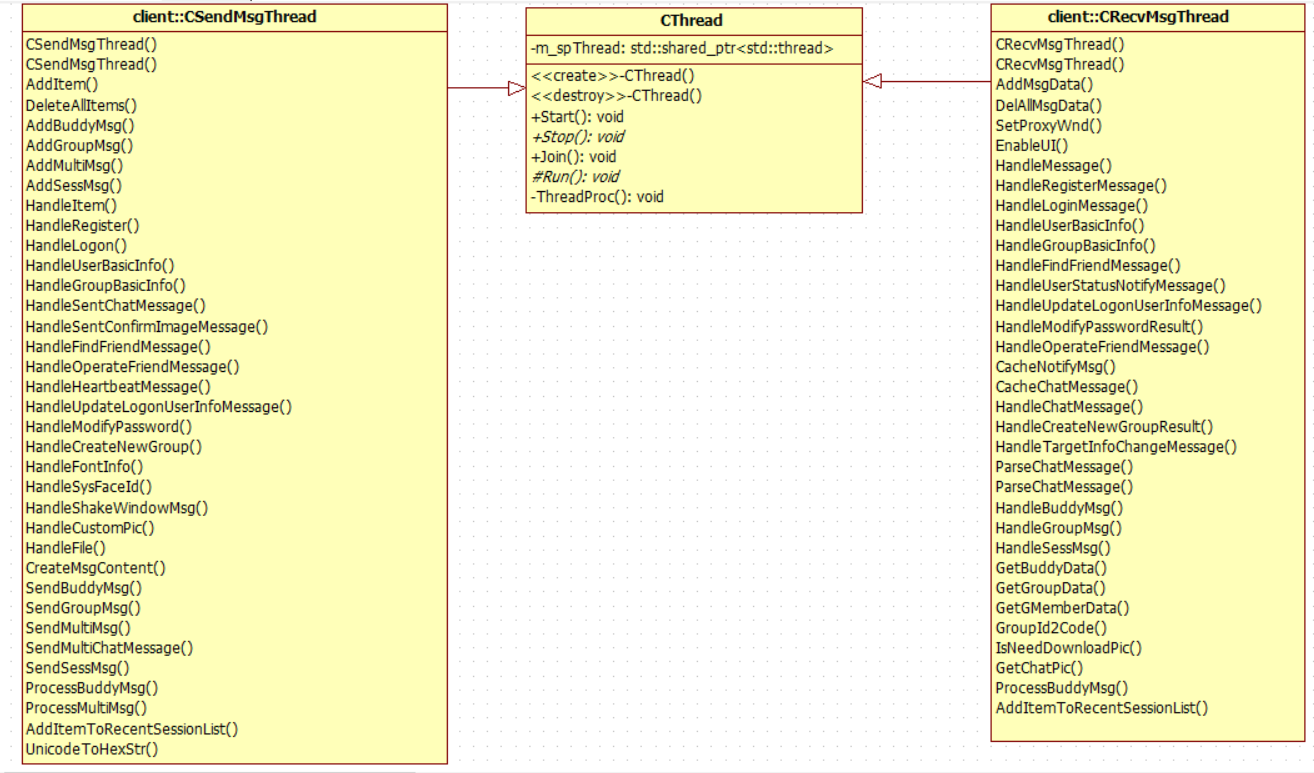


临界区域 仅在window有，效率是比互斥体高的，由系统来控制的

互斥体 无视平台，在Linux下面效率更高

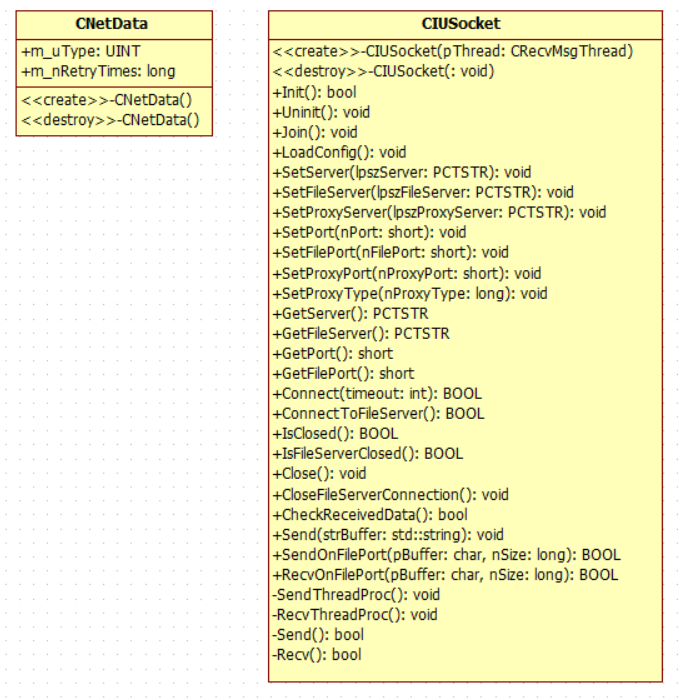
原子操作 保护的是操作

#### 消息系统的设计与实现



#### 用户界面的代码设计与实现

#### 网络模块的设计与实现



#### 数据库模块的设计与实现

采用业务模式

#### 业务模块的设计与实现

Windows消息处理与通知

业务的含义：

1 是以UI为业务（例如：聊天工具）消息处理和消息通知

2 是以功能为业务（例如：词典）

3 是依赖于服务器（搜索工具桌面版）

## 代码精讲

### 项目的搭建与编译

WTL对话框项目的创建

### 成品代码详解与调试

见代码

复盘：

1 观察现象，文字乱码

2 复现bug，观察复现的条件

3 分析情况，设置观察变量，缩小排查范围

4 找到发生问题的变化节点

5 观察问题产生的瞬间，变量的变化

6 bug可能是多种原因导致的，排除其中之一，并不能彻底解决，需要进一步分析

6 尽可能收集各种出问题的情况（编辑框、文本框显示叠字，刷新这个功能可能有问题）

### 项目的测试

见课程

## 项目总结

学习的阶段：

不知道→知道 看书、上课

知道→运用 练习（动手写代码）

运用→改进（此时才能叫做掌握）

改进→创造

1 任何复杂的项目，都是由一些基本的知识点集合而成

1. 需要由MFC开发基础
2. ATL和WTL（能够实现自定义控件）
3. Json、xml、protobuf这样的数据格式
4. 映射表的应用（图片加载、消息的映射、函数的映射）
5. Sqlite3（增删改查基本要求）
6. 调试技能和技巧（日志、分析、断点）
7. 设计这块没有定式，因地制宜，不断调整和变化