JsonCpp 部署方法：

在 http://sourceforge.net/projects/jsoncpp/中下载最新版本的 jsoncpp 库源码。
之后将 jsoncpp-src-版本号-tar.gz 解压出来，打开 makefiles 中的 jsoncpp.sln 进行编译，之后 build 文件夹下的 vs71\debug\lib_json 中会有一个.lib 静态链接库。

JsonCpp 主要包含三种类型的 class:Value Reader Writer。
jsoncpp 中所有对象、类名都在 namespace json 中，包含 json.h 即可

1. Json::Value ：可以表示所有支持的类型，如：int , double ,string , object 等
2. Json::Reader ：将文件流或字符串创解析到 Json::Value 中，主要使用 parse 函数。
3. Json::Writer ： 与 JsonReader 相反，将 Json::Value 转换成字符串流等。

注意:Jsoncpp 的 Json::Writer 类是一个纯虚类，并不能直接使用。在此我们使用 Json::Writer 的子类：Json::FastWriter、Json::StyledWriter、Json::StyledStreamWriter。

反序列化 Json（解析 Json）

?

```
{
    "name": "xiaoming",
    "like": [
        {
            "book": "json"
        },
        {
            "food": "apple"
        },
        {
            "music": "sdds"
        }
    ]
}

void ReadJson()
{
    std::string strValue = "{\"name\":\"xiaoming\",\"like\":[{\"book\":\"json\"},{\"food\":\"ap

    Json::Reader reader;
    Json::Value value;

    if (reader.parse(strValue, value))
    {
        std::string out = value["name"].asString();
        std::cout << out << std::endl;
        const Json::Value arrayObj = value["like"];
        for (unsigned int i = 0; i < arrayObj.size(); i++)
```

```
        {
            if (!arrayObj[i].isMember("book"))
                continue;
            out = arrayObj[i]["book"].asString();
            std::cout << out;
            if (i != (arrayObj.size() - 1))
                std::cout << std::endl;
        }
    }
}
```

序列化 Json(生成 Json):

?

```
void WriteJson()
{
    Json::Value root;
    Json::Value arrayObj;
    Json::Value item;

    item["food"] = "apple";
    item["music"] = "JZhou";
    item["book"] = "json";
    arrayObj.append(item);

    root["name"] = "xiaoming";
    root["like"] = arrayObj;

    root.toStyledString();
    std::string out = root.toStyledString();
    std::cout << out << std::endl;
}
```

通过 JSON 方式的 socket 传输

1、客户端：

?

```
#include "json//json.h"
#include <WinSock2.h>

#pragma comment(lib, "WS2_32.lib")
#pragma comment(lib, "json_vc71_libmtd.lib")

int main()
{
    Json::Value val;
```

```
    Json::StyledWriter style_write;

    val["name"] = "xiaoli" ;

    WSADATA wsaData;
    SOCKET SendSocket;
    sockaddr_in RecvAddr;
    int Port = 27015;

    //初始化 Socket
    WSAStartup(MAKEWORD(2, 2), &wsaData);
    //创建 Socket 对象
    SendSocket = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
    //设置服务器地址
    RecvAddr.sin_family = AF_INET;
    RecvAddr.sin_port = htons(Port);
    RecvAddr.sin_addr.s_addr = inet_addr("127.0.0.1");

    std::string SendBuf = style_write.write(val);

    //向服务器发送数据
    sendto(SendSocket, SendBuf.c_str(), SendBuf.size(), 0, (SOCKADDR*)&RecvAddr, sizeof(RecvAdd

    closesocket(SendSocket);
    WSACleanup();

    getchar();
    return 0;
}
```

2、服务器端：

?

```
#include <iostream>
#include <WinSock2.h>
#include "json/json.h"

#pragma comment(lib, "WS2_32.lib")
#pragma comment(lib, "json_vc71_libmtd.lib")
int main()
{
    //初始化 socket
    WSADATA wsaData;
    WSAStartup(MAKEWORD(2,2), &wsaData);
    //创建 socket
```

```
    SOCKET RecvSocket;
    RecvSocket = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
    //设置服务器地址
    sockaddr_in RecvAddr;
    int Port = 27015;
    RecvAddr.sin_family = AF_INET;
    RecvAddr.sin_port = htons(Port);
    RecvAddr.sin_addr.s_addr = htonl(INADDR_ANY);
    //绑定 socket
    bind(RecvSocket, (SOCKADDR*)&RecvAddr, sizeof(RecvAddr));

    char RecvBuf[1024];
    int BufLen = 1024;
    sockaddr_in SenderAddr;
    int SendAddrSize = sizeof(SenderAddr);

    recvfrom(RecvSocket, RecvBuf, BufLen, 0, (SOCKADDR*)&SenderAddr, &SendAddrSize);

    std::string strName;
    Json::Value val;
    Json::Reader reader;
    if (reader.parse(RecvBuf, val))
    {
        strName = val["name"].asString();
    }

    std::cout << strName << std::endl;

    closesocket(RecvSocket);
    WSACleanup();

    getchar();
    return 0;
}
```