# Introduction to Lib-Xtrovert

## General

Features

Cross-Platform( Linux, MS Windows )

时间：变速齿轮，虚拟的时间系统

内存管理：

## Architecture Levels

Level maps:

|  |  |
| --- | --- |
| Levels | Description |
| Application |  |
| Industry | Framework |
| Components | Foundation/Data Structure  Algorithm  (Cross -Platform) |
| Function/sys-calls encapsulation | API/Sys-Call |
| Win/Linux Function Definitions |
| Data Types | Win/Linux Data Types  wchar\_t,SBCS and MBCS |
| OS & Compiler  32/64 bits |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Levels | Description | Modules | | | | |
| Common | Utilities | Networks | IoModel | File-System |
| Application |  |  |  |  |  |  |
| Industry | Framework |  |  | NetServer/NetClient  NetReactor/Acceptor/Connector/ UdpMessager | Reactor  EventHandler | Config-File |
| Components | Foundation/Data Structure  Algorithm  (Cross -Platform) |  | Queue/Pool | TCP/UDP Socket | Linux/Windows IO Operation | Bin-File，Text-File |
| Function/sys-calls encapsulation | API/Sys-Call | xtrovert.h（common sys-header files） |  | ExtSktAPI.h | IoMoelAPI.h | File-code(UTF-8,Asc,UNICODE) |
| Win/Linux Function Definitions | Define.h  syscdef.h(sys-call def) |  | NetDefine.h |  | FileDefine.h |
| Data Types | Win/Linux Data Types  wchar\_t,SBCS and MBCS | Types.h  xchar.h(wide char) |  |  |  |  |
| OS & Compiler  32/64 bits | platform.h |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Levels | Description | Modules | | | | |
| String | Thread/Process |  |  |  |
| Application |  |  |  |  |  |  |
| Industry | Framework |  |  |  |  |  |
| Components | Foundation/Data Structure  Algorithm  (Cross -Platform) |  | Lock/Mutex |  |  |  |
| Function/sys-calls encapsulation | API/Sys-Call |  |  |  |  |  |
| Win/Linux Function Definitions |  |  |  |  |  |
| Data Types | Win/Linux Data Types  wchar\_t,SBCS and MBCS |  | Pid\_t |  |  |  |
| OS & Compiler  32/64 bits |  |  |  |  |  |

## Core System

### 3 features over an application

一个Application，可以是Server和Client的合并体，即，对某些Application来说，它是Server，同时，对另外一些Application来说，它又是Client，即，它是一个集群中的一分子。

|  |  |  |
| --- | --- | --- |
| **Terminology** | **标识** | **Description** |
| Application | 可执行程序 | 实体程序 |
| Server | 可执行程序 | 实体程序，IsKindof(Application)，存储服务器资源，用户数据，响应用户的请求，与其他Server/Client通信 |
| Client | 可执行程序 | 实体程序，IsKindof(Application)， |
| AppServer |  | 服务端应用逻辑，管理 服务，存储，通信 |
| AppClient |  | 客户端应用逻辑，管理 ServiceProxy，本地存储，通信 |
| Service |  | Computing， 面向User |
| User | UserID | 服务端的用户标识，属于存储模块的一部分 |
| Net Server |  | 管理对象是Net Client，属于通信模块 |
| Net Client | IP:Port | 属于通信模块 |

**Storage:** resource, 进程空间中的数据，内存数据,

**Communication:** 抽象IO, (IPC, networks)，(Serialize: files, configuration, static-tables, database), 不同于操作系统的IO定义，对（IO中）I的定义为，应用程序接收到的指令（网络消息，文件读入信息），对（IO）中的O的定义，应用程序运算后的结果输出数据。

**Computing:** Service, (依赖通信、存储-resource驱动)，面向User

面向服务开发

7×24小时，内存管理

方便调试，日志，axtrace输出

内建http服务：服务器统计数据收集，health（performance like memory, cpu, thread-count，网络吞吐量），networks(server的个数-以及携带的客户端个数,客户端的个数)，

客户端：运算（Service-Proxy），User-Interface-UI，通信

Communication----Storage----Service----communication

应用：

Class CExtNetSvr

ListClient( tchar \*szClientAddr ); /\* Get all active clients ip:port\*/

QuerySvrInfo(); /\* TCP/UDP, local\_addr \*/

SetIOBlock( bool bBlock );

KickClient( client\_id );

BroadCast( msg );

GroupClientCast( msg, client\_list ); /\*组播\*/

网络库：实现消息包的自动收发

连接管理：接受新连接，删除失效的连接，remote连接管理，连接存活状态

消息的收发，消息队列管理

扩展：多线程处消息队列

扩展：可复用，可被继承，进一步实现其他的扩展

逻辑消息包(Message)分类(data structure delivered from/deliver to network-layer)：

Sync: Notify my status to peer, to let peer know my status, or some other information that peer should know

Request: peer should answer result to me, whether peer proved or not.

Allow me to do something

Give me something

Control: My decision to requests that I have sent to peer

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Message-Type | Sub-type | parameters | Value | description |
| Sync | CMD\_SYNC |  |  |  |  |
| Request | CMD\_REQUEST |  |  |  |  |
| Answer | CMD\_REQ\_ANSWER |  | REQ\_SN |  |  |
| Forward |  |  |  |  | Forward payload to specified host |
| Command-Ctrl | CMD\_CTRL | Abort/ Cancel | REQ\_SN |  | 与逻辑无关的操作。  对“我”曾经send出去的命令，执行某些操作 |
| Pause/Resume | REQ\_SN |  |  |
| Re-send | REQ\_SN |  | “我”发出的请求，没有收到回应（或者回应数据有问题），或者“你”给我的消息有问题 |
| Command-Ctrl  -Answer | CMD\_CTRL\_ANSWER | - | REQ\_SN | SUCCESS | 0 to indicate success |
| - | REQ\_SN | FAILED | >0 to indicate certain error |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

网络层可以一次发送/接受多个协议包，但所有的协议包，必须是完整的

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 消息类型 | 消息长度 | 消息 | 消息类型 | 消息长度 | 消息 |
|  | Unsigned int |  |  | Unsigned int |  |
| Packet Paload[0] Packet Paload[1] ~ Packet Paload[n] | | | | | |

网络层消息包：Packet Structure

|  |  |  |  |
| --- | --- | --- | --- |
| 包头（2字节） | Payload size（2字节） | Payload | 包尾 |
| 0xAA55 | Unsigned short |  | 0x55AA |

网络层消息包：Packet Structure with fragment

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 包头（2字节） | Payload size（2字节） | 消息偏移 | Payload | 包尾 |
| 0xAA55 | Unsigned short |  |  | 0x55AA |

数据流统计(DEBUG)：

消息包时间戳(毫秒)：

|  |  |  |  |
| --- | --- | --- | --- |
| 逻辑发送 | 网络发送 | 网络接收 | 逻辑接收 |
| 源 | 源 | 目的 | 目的 |

服务端单Tick消息包计数

Reactor为处理框架，EventHandler完成事件的响应，通过对EventHandler继承，完成扩展事件处理

Acceptor: 接受连接，

Connector: 连接，消息的收发，异常处理（自己失效时）

**CNetManager**

Packet Que

Native、Peer：

Status/Information:Address Info

Operation：KickPeer，Filter(Disable,Enable,Ignore，Resume Peer), ShutdownNative

**NetReactor**

**Registered Event Handler Pointers**

**UDP Event Handlers**

Connector Handler

New

Connector Handler

**NetManager**

**Message**

**Queue**

**Connected Address**

**Pool**

**Message**

**Queue**

**Connector Handler**

**Pointers**

Acceptor Handler

**TCP Event Handlers**

**NetReactor**

**Registered Event Handler Pointers**

**Event Handlers**

Connector Handler

New

Connector Handler

**NetManager**

**Message**

**Queue**

**Connected Address**

**Pool**

**Message**

**Queue**

**Connector Handler**

**Pointers**

Acceptor Handler

运算

Storage存储

Serialize(tables, DB)

Communicate通讯

服务一组接口

Service Interface

服务逻辑

Service

Server

服务代理

Service Proxy

Communicate通讯

Client

Connector:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Init | HandleEvents |  |
| TCP Connect | 创建socket，设置服务器地址  设置事件 | EV\_READ, EV\_TIMEOUT, EV\_PERSIST | 连接服务器，定时重连  接收消息，发送消息 |
| TCP Connected |  |  |  |
| UDP Connect |  |  |  |
| UDP Connected |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

AcceptUdp:

业务(逻辑)：服务

技术：消息包

NetServer：监听1～n个端口，响应客户端服务请求的通讯组件

应用服务器AppServer：

Service：服务，应用服务器所提供的一组服务接口，每一个服务是一系列通讯数据包处理（谁who，请求了什么what服务）

Attack( CRole \*pRole );

TalkTo( CRole \*pRole, TCHAR \*szMsg );

Drive();

应用客户端AppClient：

Ac.download( filename )

{

If ( rmt\_svc.isvalid() )

Rmt\_svc.get\_file( filename );

}

Svc\_proxy.post\_request( request );

Svc\_proxy.get\_answer( request, answer );

Ac.Attack( CRole \*pRole )

{

Svc\_proxy.post\_request( req\_attack );

}

UI:

网络包延时：接收端发送网络消耗时间，网络层接收时间，服务器处理时间，网络层发送时间，接收端网络接受时间，接收端逻辑处理时间

网络层：

Receive：每次recv的数据，通知逻辑层，[起始地址,长度]

Send:每次Send，通知逻辑层，[发送字节数]

逻辑层：封包，解包

Send: 封包，copy bytes into网络层发送缓冲区

记录送给网络层的数据字节数，网络已发送的字节数。

|  |  |  |
| --- | --- | --- |
| Acceptor | | |
|  | TCP | UDP |
| Parameters | Local Address | Local Address |
| Implementation  Extension | Address Filter | Address Filter |
| Mechanism | Connection-Pool(Max-Count) | Connection-Pool(Max-Count) |
| Init | CreateListener(); | 如果检测到多块网卡，则(Local Address)==ADDR\_ANY警告  CreateListener(); |
| HandleEvents | AllocConnector();  AcceptConnector();  AddConnector();  RegisterConnector(); | AllocConnector();  RecvMessageOwner();  IsExistConnector();  If ( no )  {  AddConnector();  }  Connector->HandleEvents(); |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| Connector | | |
|  | TCP | UDP |
| Parameters | Peer Address | Local Address(Multiple-NetAdapters)  Peer Address |
| Implementation  Extension | Packet Filter  Pack/Unpack  Crypto  Zip/Unzip | Packet Filter  Pack/Unpack  Crypto  Zip/Unzip |
| Mechanism |  |  |
| Init | ConnectServer(); | CreateSocket();  SetServerSockAddr(); |
| HandleEvents | RecvMessage();  SendMessage(); | RecvMessage();  SendMessage(); |

udp:

不关注对端网络状态：在线，下线

通信不频繁：N次处理后（1分钟，几十秒？），长时间不会再处理（5分钟，或者N小时）。

LAN广播/多播消息

通信频繁，但是丢包无所谓，或者通知消息，不需要对方应答

服务器端：管理大量的客户连接数，可以避免网络吞吐量占用过高（tcp管理的socket有限，且占用过多的系统资源），此种情况，需要考虑服务器接收频繁数据处理请求的容载能力

P2P应用

tcp：

关注对端网络状态

通信频繁，且数据要确保到达对端

数据发送安全，确保到达对端

服务器端：客户端连接，不是海量，或者大量的客户端管理（<1w?）

## Log System

TEXT\_LOG, TXT\_LOG， 文本log文件，for human reading，记录上层逻辑处理过程，用于统计、分析业务数据，或者初步的bug定位

RAW\_LOG 二进制log文件, for program reading，记录程序的输入、输出数据流，程序直接加载log文件，重现程序的数据处理过程，用于调试、分析上层应用程序的bug，crash

默认的logger是 printf

网络logger初始化前，或者初始化失败，logger自动初始化为文件。

Logger文件初始化失败，则自动初始化为printf

CExtLogger提供日志输出的可扩展性，

CExtLoggerBackend 参与日志系统的具体实现

class CExtLogger (Proxy)

{

public:

virtual int Init(void);

virtual int Write( const TCHAR \*szLogMsg, size\_t nLen );

virtual int WriteStr( const TCHAR \*szLogMsg );

protected:

virtual int GetTime(void);

private:

};

class CExtLoggerBackend : public CExtLogger

{

public:

protected:

private:

};

LogServer

Logger (Client)

Backend Ingerface (Client)

CExtLoggerBkendFile

LOG\_BKEND\_DB

LOG\_BKEND\_NET

LOG\_BKEND\_CACHE

Dump(VOID)

CExtLogFile

Sierializer

## Utilities

|  |  |  |
| --- | --- | --- |
| CXtObjPool |  | |
|  | Functions | Description |
| Public |  |  |
|  |  |  |
| Protected |  |  |
| Private |  |  |
|  |  |  |
| Structure:  **Object Handle Buffer**  **Capacity**  **Real Capacity**  Capacity NULL  Count  **Memory Chunks**  **NULL**  m\_nRealCapacityGrow  **Object Pool** | | |

## Code Tree

|  |  |
| --- | --- |
| **一级目录** | **说明** |
| ext\_api | 紧对 平台、数据类型、系统调用、C函数 等进行封装，该目录可以脱离extrovert lib库单独使用 |
| gallery | 第三方，以源码发布的（非lib库房是使用的），公开的算法等，算法体积小  实用工具和代码，仅基于ext\_api开发，该目录可以与ext\_api目录联合，脱离extrovert lib库单独使用 |
| inc | extrovert lib库基础头文件 |
| src | extrovert lib库基础实现 |
| utility | extrovert lib库使用的组件，数据结构等 |

低级目录

|  |  |
| --- | --- |
| **Gallery目录** | |
| **目录** | **文件列表/说明** |
| 根目录 | 对internal目录中的文件，封装，方便后续替换  digest\_file : 文件摘要信息（md5码），处理。 |
| internal | 内部实现方式，不直接使用该目录的文件 |
| 3rd\_part | Md5 |
|  |  |
|  |  |

## Error Code Categary

|  |  |
| --- | --- |
| **General Function Returned Errors** | |
| **Code** | **Description** |
| FUN\_RET\_OK , 0 | Function returned normally, with NO error occurs. |
| FUN\_RET\_ERR\_UNKNOWN, -1 | Function returned unknown error, need errno to get further information. |
| FUN\_RET\_ERR\_ILLEGAL\_ARGS, -2 | Illegal parameters passed into function. |
| FUN\_RET\_ERR\_ILLEGAL\_CALL, -3 | Causes illegal operations while calling this function. Obsoleted. |
| FUN\_RET\_ERR\_NOTIMPL, -4 | Function or method NOT supported. |

|  |  |
| --- | --- |
| **System Errors** | |
| **Code** | **Description** |
| 内存 |  |
| 网络 | IO error |
| 文件 | IO error |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| 用户自定义 |  |

|  |  |
| --- | --- |
| **Application Errors** | |
| **Code** | **Description** |
| 内存 |  |
| 网络 |  |
| 文件 |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| 用户自定义 |  |