**Fossilized**

fossilizid: a open source c++ library start by wmn0377[@corp](http://git.mt.nie.netease.com/user/corp).netease.com

## container: lock-free structure

fossilizid::container::msque

基于单链表的无锁队列

相关论文<http://www.research.ibm.com/people/m/michael/podc-1996.pdf>

<http://web.cecs.pdx.edu/~walpole/class/cs510/papers/11.pdf>

optimisticque

基于双链表的对锁队列

相关论文<https://www.offblast.org/stuff/books/FIFO_Queues.pdf>

ringque

基于定长数组实现的环形队列

swapque

基于读写锁，队列本身包含2个子队列，一个用于push，一个用于pop

队列采用了统一的接口设计

*bool empty()*

*判断队列是否为空,空返回true否之返回false*

*std::size\_t msque::size()*

*获取队列长度,返回当前队列元素数目*

*void msque::clear()*

*清空队列*

*void msque::push(const T & data)*

*将元素插入队列*

*bool msque::pop(T & data)*

*将元素弹出队列*

small\_hash\_map

基于读写锁的hash\_map，对bucket进行加锁

Interface

*void for\_each(boost::function<void(V var) > handle )*

*遍历hash\_map*

*bool set(K key, V value)*

*设置对应key的value*

*void insert(K key, V value)*

*插入（key，value）*

*bool search(K key, V &value)*

*查找指定key*

*bool erase(K key)*

*删除指定key*

*unsigned int size()*

*获取hash\_map的元素数目*

例子: fossilizid/test/test\_container

## pool: mempool&&objpool

fossilizid::pool::mempool

内存池，按分配的内存大小做了简单的分支管理，小于64K的内存采用链表管理，在新的内存块上保存上级节点的指针地址，大于64K的内存采用红黑树保存，直接采用了std::map

Interface

*static void \* allocator(int len)*

*分配内存*

*static void deallocator(void \* buff, int len)*

*回收内存*

fossilizid::pool::factory

对象池，采用可变长模板参数适配不同参数的构造函数

Interface

*template<class T, typename ...Tlist>*

*static T \* create(int count, Tlist&& ... var)*

*创建count个数的对象*

*template<class T, typename ...Tlist>*

*static T \* create(Tlist&& ... var)*

*创建一个对象*

*template<class T>*

*static void release(T \* p, int count)*

*释放count个对象*

例子: fossilizid/ test/test\_pool

## remoteq: network library

fossilizid::remoteq

基于模板适配网络协议

Interface

*ACCEPTOR acceptor(QUEUE que, ENDPOINT ep)*

*创建接收器*

*CHANNEL accept(ACCEPTOR ap)*

*接收接入的CHANNEL*

*CHANNEL connect(ENDPOINT ep, QUEUE que = 0)*

*接入远端*

*void close(HANDLE \_handle)*

*释放句柄*

*ENDPOINT endpoint(char \* ip, short port)*

*创建地址*

*QUEUE queue()*

*创建事件队列*

*EVENT queue(QUEUE que)*

*获取事件*

例子: fossilizid/test/test\_remote\_queue

## reliablyt: udp reliably transmission

reliablyt

基于停等协议的udp可靠性传输

Interface

*class UDPSession{*

*public:*

*boost::signals2::signal<void(char \*, int) > sigRecv;*

*boost::signals2::signal<void() > sigDisConnect;*

*void disconnect();*

*void reliable\_send(char \* buf, int len);*

*void unreliable\_send(char \* buf, int len);*

*}*

*class UDPService : public UDPBase{*

*public:*

*boost::signals2::signal<void(boost::shared\_ptr<UDPConnect>) > sigConnect;*

*}*

例子: fossilizid/test/test\_udp

## reduce: service

fossilizid::reduce

基于remoteq及context的service,支持阻塞式rpc

Interface

*class acceptservice;*

*监听service*

*class connectservice;*

*接受service*

*class locale\_obj;*

*本地obj*

*class remote\_obj;*

*远程obj*

例子: fossilizid/test/ test\_service

## vchat: voice chat framework

vchat

基于portaudio，speex的多人语音聊天框架

Interface

*class paInit{*

*public:*

*paInit();*

*~paInit();*

*};*

*初始化 portaudio*

*class devices{*

*public:*

*static std::vector<const PaDeviceInfo\*> getInputDevices();*

*static std::vector<const PaDeviceInfo\*> getOutputDevices();*

*};*

*获取设备列表*

*class encode{*

*public:*

*int encoded(char \* inbuf, int framelen, char \* outbuf, int outbuflen);*

*int decoded(char \* inbuf, int framelen, char \* outbuf, int outbuflen);*

*int getframesize();*

*};*

*编解码器*

*class sound{*

*public:*

*void start();*

*void stop();*

*boost::signals2::signal<void(char \*, int)> sigCapture;*

*bool setOutputDevice(PaDeviceIndex index);*

*bool setInputDevice(PaDeviceIndex index);*

*void setsoundsize();*

*void setechostate(bool on);*

*};*

*采集接口*

*boost::signals2::signal<void(char \*, int)> sigCapture*

*采集音频回调*

*struct client{*

*bool read\_buff(char \* & outputbuff, short & channelcount, int &len);*

*void write\_buff(char \* buff, int buflen, short channelcount);*

*};*

*接入聊天的用户,用于缓存该用户的语音数据*

*client \* create\_client(int index = 0)*

*创建用户*

*client \* get\_client(int index);*

*获取用户*

*typedef void(\*handle\_iterator\_client)(std::map<int, client\*> & set)*

*void iterator\_client\_set(handle\_iterator\_client fn);*

*void iterator\_client\_set(std::function<void(std::map<int, client\*> &) > fn);*

*遍历用户*

*bool destroy\_client(int index);*

*删除用户*

*int client\_count();*

*获取用户数目*

例子: fossilizid/test/ test\_vchat