

# Fossilized

---

fossilizid: a open source c++ library start by wmn0377@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](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** *deallocater*(*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

---

## remote-queue: network library

fossilizid::remote\_queue

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

---

## reliably-transmission: udp reliably transmission

reliably-transmission

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

---

# vchat: voice chat framework

## vchat

基于 portaudio, speex 的多人语音聊天框架

## Interface

```
class palnit{
public:
    palnit();
    ~palnit();
};
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 setsoundszise();
    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

---

