链滴

一个轻量级的跨平台c开发库：TBOX

作者：waruqi

原文链接：https://ld246.com/article/1466933213630

来源网站：链滴

简介

TBOX是一个用c语言实现的多平台开发库，支持windows、linux、mac、ios、android以及其他嵌式系统。

针对各个平台，封装了统一的接口，简化了各类开发过程中常用操作，使你在开发过程中，更加关注际应用的开发，而不是把时间浪费在琐碎的接口兼容性上面，并且充分利用了各个平台独有的一些特进行优化。

- 在线文档
- 在线源码

流库

针对http、file、socket、data等流数据，实现统一接口进行读写，并且支持: 阻塞、非阻塞、异步种读写模式。

支持中间增加多层filter流进行流过滤，实现边读取，内部边进行解压、编码转换、加密等操作，极大减少了内存使用。

主要提供以下特性：

1. stream：通用非阻塞流，用于一般的单独io处理。
2. async_stream：利用asio实现的纯异步流，基于回调模式，可同时处理大量并发io。
3. transfer：传输器，维护两路流的传输，对async_stream的使用进行更上层的封装，用其可以很方的实现下载、上传、复制等io传输操作。
4. transfer_pool：传输池，基于asio，维护大量并发的传输，可以用于实现爬虫、批量下载等等。
5. static_stream：针对静态数据buffer优化的静态流，用于轻量快速的数据解析。

asio库

1. 支持reactor和proactor两种模型，针对不同平台，采用epoll/poll/select/kqueue/iocp接口，最化异步操作的性能。
2. 并且对http、ssl、dns也提供了纯异步模式的实现。基于此库完全可以很方便的写出一个高性能的型服务器。

数据库

1. 统一并简化数据库操作接口，适配各种数据源，通过统一的url来自动连接打开支持的数据库，数的枚举采用迭代器模型。
2. 目前支持sqlite3以及mysql两种关系型数据库，也可自定义扩展使用其他关系型数据库。

xml库

1. 针对xml提供DOM和SAX两种解析模式，SAX方式采用外部迭代模式，灵活性和性能更高，并且可选择指定路径，进行解析。
2. 解析过程完全基于stream，所以是高度流化的，可以实现边下载、边解压、边转码、边解析一条

服务，使用较低的内存也可以解析大规模数据。

3. 提供xml writer以支持对xml生成

内存库

1. 参考linux内核内存管理机制的实现，并对其进行各种改造和优化，所实现的TBOX独有的一整套内池管理架构。

2. 调试模式下，可以轻松检测并定位内存泄露、内存越界溢出、内存重叠覆盖等常见内存问题，并对体内存的使用进行了统计和简要分析。

3. 针对大块数据、小块数据、字符串数据进行了充分的利用，避免了大量外部碎片和内部碎片的产生分配操作进行了各种优化，96%的情况下，效率都是在O(1)。

容器库

1. 提供哈希、链表、数组、队列、堆栈、最小最大堆等常用容器。

2. 支持各种常用成员类型，在原有的容器期初上，其成员类型还可以完全自定义扩展。

3. 所有容器都支持迭代器操作。

4. 大部分容器都可以支持基于stream的序列化和反序列化操作。

算法库

1. 提供各种排序算法：冒泡排序、堆排序、快速排序、插入排序。

2. 提供各种查找算法：线性遍历、二分法搜索。

3. 提供各种遍历、删除、统计算法。

4. 以迭代器为接口，实现算法和容器的分离，类似stl，但是c实现的，更加轻量。

网络库

1. 实现http、cookies、dns解析与缓存、ipv4、url的封装。

数学运算库

1. 提供各种精度的定点运算支持

2. 提供随机数生成器

libc库

1. libc的一个轻量级实现，完全跨平台，并且针对不同架构进行了优化。

2. 支持大部分字符串、宽字符串操作。

3. 扩展字符串、宽字符串的各种大小写不敏感操作接口

4. 扩展memset_u16、memset_u32等接口，并对其进行高度优化，尤其适合图形渲染程序

libm库

1. libm部分接口的一个轻量级实现，以及对常用系统接口的封装。（目前只实现了部分，之后有时间完全实现掉）

2. 扩展部分常用接口，增加对sqrt、log2等常用函数的整数版本计算，进行高度优化，不涉及浮点运，适合嵌入式环境使用。

object库

1. 轻量级类apple的CoreFoundation库，支持object、dictionary、array、string、number、date data等常用对象，并且可以方便扩展自定义对象的序列化。

2. 支持对xml、json、binary以及apple的plist(xplist/bplist)格式序列化和反序列化。

并且实现自有的binary序列化格式， 针对明文进行了简单的加密，在不影响性能的前提下，序列化后大小比bplist节省30%。

平台库

1. 提供file、directory、socket、thread、time等常用系统接口

2. 提供atomic、atomic64接口

3. 提供高精度、低精度定时器

4. 提供高性能的线程池操作

5. 提供event、mutex、semaphore、spinlock等事件、互斥、信号量、自旋锁操作

6. 提供获取函数堆栈信息的接口，方便调试和错误定位

7. 提供跨平台动态库加载接口（如果系统支持的话）

压缩库

1. 支持zlib/zlibraw/gzip的压缩与解压（需要第三方zlib库支持）。

字符编码库

1. 支持utf8、utf16、gbk、gb2312、uc2、uc4 之间的互相转码，并且支持大小端格式。

实用工具库

1. 实现base64/32编解码

2. 实现crc32、adler32、md5、sha1等常用hash算法

3. 实现日志输出、断言等辅助调试工具

4. 实现url编解码

5. 实现位操作相关接口，支持各种数据格式的解析，可以对8bits、16bits、32bits、64bits、float、ouble以及任意bits的字段进行解析操作，并且同时支持大端、小端和本地端模式，并针对部分操作行了优化，像static_stream、stream都有相关接口对其进行了封装，方便在流上进行快速数据解析。

6. 实现swap16、swap32、swap64等位交换操作，并针对各个平台进行了优化。

7. 实现一些高级的位处理接口，例如：位0的快速统计、前导0和前导1的快速位计数、后导01的快速计数

8. 实现单例模块，可以对静态对象、实例对象进行快速的单例封装，实现全局线程安全

9. 实现option模块，对命令行参数进行解析，提供快速方便的命令行选项建立和解析操作，对于写终程序还是很有帮助的

正则表达式库

1. 支持匹配和替换操作

2. 支持全局、多行、大小写不敏感等模式

3. 使用pcre, pcre2和posix正则库

一些使用tbox的项目：

- gbox
- itrace
- 更多项目

编译

请先安装: xmake

```
// 默认直接编译当前主机平台
cd ./tbox
xmake

// 编译iphoneos平台
cd ./tbox
xmake f -p iphoneos
xmake

// 编译android平台
cd ./tbox
xmake f -p android --ndk=xxxxx
xmake
```

例子

```
#include "tbox/tbox.h"

int main(int argc, char** argv)
{
    // init tbox
    if (!tb_init(tb_null, tb_null)) return 0;

    // trace
    tb_trace_i("hello tbox");

    // init vector
```

```
    tb_vector_ref_t vector = tb_vector_init(0, tb_element_cstr(tb_true));
    if (vector)
    {
        // insert item
        tb_vector_insert_tail(vector, "hello");
        tb_vector_insert_tail(vector, "tbox");

        // dump all items
        tb_for_all (tb_char_t const*, cstr, vector)
        {
            // trace
            tb_trace_i("%s", cstr);
        }

        // exit vector
        tb_vector_exit(vector);
    }

    // init stream
    tb_stream_ref_t stream = tb_stream_init_from_url("http://www.xxx.com/file.txt");
    if (stream)
    {
        // open stream
        if (tb_stream_open(stream))
        {
            // read line
            tb_long_t size = 0;
            tb_char_t line[TB_STREAM_BLOCK_MAXN];
            while ((size = tb_stream_bread_line(stream, line, sizeof(line))) >= 0)
            {
                // trace
                tb_trace_i("line: %s", line);
            }
        }

        // exit stream
        tb_stream_exit(stream);
    }

    // wait some time
    getchar();

    // exit tbox
    tb_exit();
    return 0;
}
```