链滴

玩 c 的同学快进来，推荐个跨平台 c 库给你们

作者：waruqi

原文链接：https://ld246.com/article/1533173724003

来源网站：链滴

最近终于腾出时间把两年前的 issues#24给解决了，现在 windows 上已经完全支持基于 IOCP 的协程 o 处理。

大家有兴趣的话，或者有需要用 c 开发跨平台程序的同学可以关注下哦，里面有各种常用模块，并且供了灵活的裁剪编译模式，方便大家只编译使用需要的模块。

摆着代码即文档的原则，实现接口都有详细注释说明，同时 TBOX 也是个代码仓库，大家如果在写系程序的时候，一些系统接口的使用可以在 tbox 的源码的 platform 目录中搜索，找到相关实现来做考哦。

前两天刚好发了新版本(v1.6.3)，花了我一年多时间修复了各种问题，不过中途又跑去捣鼓xmake(一基于 lua 的跨平台编译工具)了，大家也可以关注下，嘿嘿。

简介

TBOX是一个用c语言实现的跨平台开发库。

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。

这个项目的目的，是为了使C开发更加的简单高效。

目前支持的平台有：

- Windows
- Macosx
- Linux
- Android
- iOS

通过xmake支持各种编译模式：

- Release: 正式版编译，禁用调试信息、断言，各种检测机制，启用编译器优化
- Debug: 调试模式，默认启用详细调试信息、断言、内存越界检测、内存泄漏、锁竞争分析等检测制
- Small: 最小化编译，默认禁用所有扩展模块，启用编译器最小化优化
- Micro: 针对嵌入式平台，仅仅编译tbox微内核，仅提供最基础的跨平台接口，生成库仅64K左右（置轻量libc接口实现）

如果你想了解更多，请参考：

- 项目主页
- 在线文档
- Github
- Gitee

特性

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。

主要提供以下模块：

- stream：通用非阻塞流，用于一般的单独io处理，同时支持协程以实现异步传输。
- transfer：流传输器，维护两路流的传输。
- 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。

协程库

- 快速高效的协程切换支持（具体性能参考： 基准测试报告）
- 提供跨平台支持，核心切换算法参考boost，并且对其进行重写和优化，目前支持架构：x86, x86_6 , arm, arm64, mips32
- 提供channel协程间数据通信支持，基于生产、消费者模型
- 提供信号量、协程锁支持
- socket、stream都模块原生支持协程，并且可在线程和协程间进行无缝切换
- 提供http、file等基于协程的简单服务器实例，只需几百行代码，就可以从socket开始写个高性能i服务器，代码逻辑比异步回调模式更加清晰
- 同时提供stackfull, stackless两种协程模式支持，stackless协程更加的轻量（每个协程只占用几十bytes），切换更快（会牺牲部分易用性）
- 支持epoll, kqueue, poll, select 和 IOCP

数据库

- 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，数据枚举采用迭代器模型。
- 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

- 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。
- 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、边解析一条龙务，使用较低的内存也可以解析大规模数据。
- 提供xml writer以支持对xml生成

内存库

- 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。

● 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。

● 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

● 提供哈希、链表、数组、队列、堆栈、最小最大堆等常用容器。

● 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。

● 所有容器都支持迭代器操作。

● 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

● 提供各种排序算法：冒泡排序、堆排序、快速排序、插入排序。

● 提供各种查找算法：线性遍历、二分法搜索。

● 提供各种遍历、删除、统计算法。

● 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

● 实现http客户端模块

● 实现cookies

● 实现dns解析与缓存

● 实现ssl(支持openssl, polarssl, mbedtls)

● 支持ipv4、ipv6

● 支持通过协程实现异步模式

数学运算库

● 提供各种精度的定点运算支持

● 提供随机数生成器

libc库

● 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。

● 支持大部分字符串、宽字符串操作。

● 扩展字符串、宽字符串的各种大小写不敏感操作接口

● 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

● 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）

● 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

● 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。

● 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

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

● 提供atomic、atomic64接口

● 提供高精度、低精度定时器

● 提供高性能的线程池操作

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

● 提供获取函数堆栈信息的接口，方便调试和错误定位

● 提供跨平台动态库加载接口（如果系统支持的话）

● 提供io轮询器，针对epoll, poll, select, kqueue进行跨平台封装

● 提供跨平台上下文切换接口，主要用于协程实现，切换效率非常高

压缩库

● 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

● 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

● 实现base64/32编解码

● 实现crc32、adler32、md5、sha1等常用hash算法

● 实现日志输出、断言等辅助调试工具

● 实现url编解码

● 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。

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

- 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
- 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
- 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

- 支持匹配和替换操作
- 支持全局、多行、大小写不敏感等模式
- 使用pcre, pcre2和posix正则库

一些使用tbox的项目：

- gbox
- vm86
- xmake
- 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mingw平台
$ cd ./tbox
$ xmake f -p mingw --sdk=/home/mingwsdk
$ 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

# 交叉编译
$ cd ./tbox
$ xmake f -p linux --sdk=/home/sdk #--bin=/home/sdk/bin
$ 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
    getchar();
```

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