翻译：http://yaocoder.blog.51cto.com/
联系方式：yaocoder@gmail.com

关于 libjson

项目主页：http://sourceforge.net/projects/libjson/

libjson 是一个适用于 C 和 C++语言高效的并且可灵活定制的 json 库。它拥有针对 C++的接口，使 C++开发人员也能够在他们的程序中使用 json，同时由于多数 libjson 的方法（函数）是内敛的，更进一步提升了库的效率。

它也为其他类型的语言提供了 C 风格的接口和共享库。文档将会讲解如何去构建、优化 libjson 来适应你的需求，并且对 C、C++接口进行详细的说明。

libjson 当前最新版本为7.*，如果你想把程序中的 libjson 5或是更早的版本升级到此版本，请注意相应的接口变化，但是也不用担心，所有的函数功能没有变化，它们只是更完善，更标准化。如果你想从 libjson 6升级，仅有的变化是 json_validate 这个接口。 libjson 的接口在升级中（完善和修正 bug）并没有发生变化，你可以查看每一次版本升级的 changelog。

Libjson 不仅高效，功能也很丰富，包括 parsers, writers, builders, formatters, validators… ，它还可以根据你的需求进行灵活的定制。也许你会感到奇怪：libjson 看上去如此轻量但是却拥有一个200多页的文档和很多文件，但事实上你可以根据你的需求只使用需要的部分。JSONOptions.h 文件中介绍了这些配置选项。在后面的部分将给你详细介绍这些配置选项的适用场景，这些选项允许你根据自己的程序去做相应的优化。libjson 设计的很精巧，它只使用少量内存，并且很高效。所以尽量不要对库本身做一些不紧要的微调。

libjson 相比其他库更具备可定制性，包括 mutex 管理，垃圾收集（garbage collection），内存管理（memory control），unicode 支持，可以灵活的定制而非只提供给你默认的行为。它拥有纯净的 C 接口，所有 C 风格的接口都有 json_ 前缀。它也拥有非常直观的 C++接口，比如迭代器（iterators），与 STL 的风格相似。C 和 C++两种库共享了部分代码，但也有各自特定的代码。

libjson 是安全的，它的默认行为就具有抵御拒绝服务攻击（DoS）的能力，并且它还拥有其他可配置的安全选项。

支持的平台：
libjson 官方支持并且进行过测试验证的是以下平台：
gcc (OSX) ——完全支持
llvm (OSX) - 完全支持
gcc (LInux) - 完全支持

MinGW (Windows) - 支持，部分经过测试
Visual C++ (Windows) - 支持，部分经过测试

构建 libjson

libjson 可被有经验的开发人员进行专业化定制，也可以被新手简单的使用。大多数用户只需要生成库文件，如果你想在你的 C++程序中使用它，你必须把依赖的源码文件加入你的工程并且注释掉 JSONOptions.h 文件中的 JSON_LIBRARY，并使用相应编译器编译成的库文件。

JSONOptions.h

在源码文件中你将发现 JSONOptions.h，它被用来进行构建前的配置工作。如果你不需要某项功能，只要把相应的宏注释掉就可以了，在文件中每个功能都会有一段简要的说明。
（注：强烈建议你在程序中使用它前最好先构建运行提供的单元测试进行下验证。由于各种选项的组合搭配太多了，并不是所有的都经过验证。）

编译选项

类 UNIX 环境下使用 makefile。libjson 的所有行为选项都在 JSONOptions 中，你可以通过在编译选项来控制它的行为。makefile 的基本选项为构建类型（BUILD_TYPE），是否为共享库（SHARED）和安装（install）。

默认运行 make 会在当前路径下生成静态库。运行 make install 将会安装在路径 exec_prefix（默认为/usr/local）但是你可以改变为其他路径。
你还可以通过设置 BUILD_TYPE 来进行其它编译，如图

	BUILD_TYPE=debug	BUILD_TYPE=small
Output	libjson_dbg.a	libjson.a
Library Options	JSONOptions.h, and JSON_DEBUG	JSONOptions.h and JSON_LESS_MEMORY
Compile Options	-g -Wall	-Os, -ffast-math -DNDEBUG

通过设置 SHARED=1可以生成动态链接库，默认 SHARED=0生成的是静态库。

JSON_LIBRARY

这个选项生成的是 C 风格接口的动态库或者静态库，可以被兼容 C 接口的多种编程语言使用，与 C++的库相比功能性相同。

推荐用途

用来生成静态库或者动态库供其他非 C++的程序使用

默认开启
是

JSON_STRICT
此选项要求必须严格遵守 JSON 标准。排除了一些非 json 标准但是 libjson 支持的,比如注释，16进制，八进制等;

推荐用途
严格遵守 json 协议的应用

要求
JSON_UNICODE 必须开启
JSON_COMMENTS 必须关闭
JSON_OCTAL 必须关闭

默认开启
否

JSON_DEBUG
此选项用于创建调试版本的库。由于它执行了很多内部检查，执行效率会有所下降，所以用在程序的 debug 版本非 release 版本。
此选项提供了接口 libjson::registerDebugCallback （ C++ ），json_register_debug_callback（C）以便出问题时得到反馈。它们只是反馈错误而并不能处理错误。如果你想把错误当做异常理需开启 JSON_SAFE 选项。你也可以开启 JSON_STDERROR 使错误通过 stderr 输出，在这种情况下回调接口失效。
此选项还提供了接口 JSONNode::dump（C++），这在调试时非常有用，你可以检查库的工作情况。C 接口不支持这个功能，。because it is mostly used by the author and maintainers。

推荐用途
用在程序的 debug 版本，不能用于 release 版本;

默认开启
否

JSON_ISO_STRICT
此选项使 libjson 严格遵守 ISO 标准。它会移除非标准 C++的用法，比如 long longs 和 wchar_t;

推荐用途
你所使用的编译器的标准

默认开启

一般情况下是关闭的，但是如果你的 gcc 启用了 ansi 选项，这个选项会自己开启并且给出警告信息。

JSON_SAFE

此选项在调试情况下使用，但是你也可以用在认为可能会接收到错误格式 json 的情况下。此选项使你在遇到错误的情况下，通过回调或者 stderr 输出错误信息（因为它开启了内部的验证功能）。

推荐用途

用在可能会接收到错误格式 json 的情况或者你想对 json 进行验证。如果你能确保要处理的 json 准确无误，这个选项不需要开启。

默认开启

Yes

JSON_CASTABLE

此选项使两个内部类型可以进行隐式转换。比如你调用 as_int 作为 bool 值，如果正确将返回1，错误返回0。如果这个选项未开启，这种行为是非法的。此选项使 libjson 更灵活。

推荐用途

使程序更灵活或者支持弱类型的情况下使用

默认开启

Yes

JSON_STDERROR

此选项使 libjson 错误通过 stderr 输出，相应的会使 ibjson::registerDebugCallback 和 json_register_debug_callback 失效。

推荐用途

调试模式下输出错误但不做其他处理的情况

默认开启

No

JSON_PREPARSE

此选项使 libjson 对整个 json 字符串进行立刻解析。在默认情况下，libjson 进行快速解析，这种情况最初解析比较快并且用了较少的内存，但是随后的读取比较慢。此选项使情况恰恰相反。

推荐用途

在 json 被大量使用在 C/S 模型的通信的情况下，server 端会发送大量的信

息，解析会耗费时间。但是，json 也可用做配置文件等情况下，在这种情况下使用此选项比较好。

默认开启
No

JSON_LESS_MEMORY
此选项使 libjson 少用了大约20%的内存。当 libjson 使用完内存后并不会完全释放，以便之后再次使用。此选项使 libjson 完全释放内存。

推荐用途
需要大量的内存的程序或者内存资源宝贵的系统，比如嵌入式系统。

默认开启
No

JSON_UNICODE
此选项使 libjson 的接口必须使用宽字符集，因为 json 标准规定必须完全支持 UTF-8。但是由于多字符集很少使用，这个选项默认是关闭的。

推荐用途
使用 UTF-8字符集的程序

要求
不能以 ansi 方式编译

默认开启
No

JSON_REF_COUNT
此选项使 libjson 的节点对象拥有引用计数（reference count）的特性，内部结构拥有写时复制（copy-on-write）的特性。这使得以传值的方式使用和拷贝节点速度很快，但是不具备线程安全。

推荐用途
开发人员大量的使用拷贝或者传值，也可以用在仅仅只是读 JSONNodes。但是如果你是在多线程的情况下使用 libjson，建议你关掉此选项，或者使用 JSON_MUTEX_CALLBACKs 或者关键区（critical sections）。

默认开启
Yes

JSON_BINARY
此选项启用了 set_binary 和 get_binary 方法。因为 json 标准未提供对

二进制数据的编码，libjson（和其他大多数 json 库一样）使用 Base64来编码解码二进制数据。使得你可以把图片或者一些文件采用二进制数据在客户端和服务端之间传输。你也可以使用第三方的 Base64库，但是 libjson 自带的 Base64是经过高度优化的应该比其他大多数第三方的库效率更高。

推荐用途

开发人员不得不对类似图片一类的文件进行二进制数据的编码解码。也可以使用此选项对数据进行模糊化处理。

默认开启

Yes

JSON_EXPOSE_BASE64

此选项提供了有关 Base64编码功能的接口。你也许想使用其他的 Base64库，但是 libjson 自带的 Base64功能是我认为比较高效的，也更适合你。另外 josn 和 Base64都经常被用在网络传输数据领域。

推荐用途

开发人员不得不对类似图片一类的文件进行二进制数据的编码解码。也可以使用此选项对数据进行模糊化处理。

默认开启

Yes

JSON_ITERATORS

此选项开启了 libjson 中有关迭代器(iterator)功能的方法。比如 erase, find, insert ，和 C++ STL 迭代器相比有诸多类似之处但也有诸多限制。

推荐用途

开发人员想通过类似于 C++ STL 迭代器功能的方法来操作 libjson。

默认开启

Yes

JSON_STREAM

此选项开启了 libjson 中有关流（ streaming）功能的方法。使你可以缓存 json，以流的方式发送。

推荐用途

开发人员想要缓存 json，或者需要网络流。

默认开启

Yes

JSON_MEMORY_CALLBACKS

此选项提供了用来进行内存分配、调整、释放的回调函数。如果用户想要进一步提高 libjson 库的效率有可能利用内存池技术，可以通过注册自定义的回调函数来实现。开启此选项如果没有注册相应的回调函数将按照库的默认行为。

推荐用途

开发人员想要使用自定义的内存处理函数，比如内存池和垃圾收集器（garbage collection）。

默认开启

No

JSON_MEMORY_MANAGE

此选项提供了用来进行大量内存释放(由 libjson 分配的内存)的功能函数，包括字符串（strings）和节点（nodes）。

推荐用途

开发人员使用了大量的 strings 和 nodes 并且需要频繁清理或者内存管理。

默认开启

No

JSON_MEMORY_POOL

此选项开启了内存池功能。这将会进一步提高 libjson 库的效率。JSON 经常被用在客户端和服务器的通信，需要解析 json，创建 json，结束回话（session）。在这种情况下需要经常进行内存的分配和回收，内存池技术可以解决这个问题。

推荐用途

开发人员需要时常解析 json。

默认开启

No

JSON_MUTEX_CALLBACKS

此选项提供了用来进行对互斥量（mutex）进行加解锁和对 JSONNOdes 及其子节点进行加解锁的回调函数。这样可以使开发人员更好对互斥量进行管理。

推荐用途

开发人员需要在竞争条件下使用 libjson。

默认开启

No

JSON_MUTEX_MANAGE

配合以上 JSON_MUTEX_CALLBACKS 功能进行回调实现。

推荐用途

开发人员需要在竞争条件下使用 libjson。

要求

JSON_MUTEX_CALLBACKS 必须被定义。

默认开启

No

JSON_NO_C_CONSTS

此选项只是移除了函数申明的 const 属性。

推荐用途

开发人员使用传统的 C 编译器和代码规范。

默认开启

No

JSON_OCTAL

此选项使得字符串和数字的值支持八进制，默认是十进制。

推荐用途

开发人员需要使用八进制。

默认开启

No

JSON_READ_PRIORITY

此选项提供了 json 读取解析函数，并且可以设置 json 解析优先级。通常读是最高优先级，但是你的应用中涉及到大量的写，你可以根据需求调整优先级来更好地适应写。

推荐用途

程序中需要大量的 json 解析。

默认开启

Yes（优先级：最高）

JSON_WRITE_PRIORITY

此选项提供了 write 和 write_formatted 函数，并且也可以设置优先级。如果不开启此选项，libjson 仅仅具有读取 json 的功能。

推荐用途

程序中需要 json 写。

默认开启

Yes（优先级：中）

JSON_NEWLINE

此选项用来指定写格式化（writing formatted ）json 时指定换行符。默认 libjson 使用的是 UNIX 换行符\n，你可以指定为 windows 下的换行符\r\n 或者基于 HTML 的</br>。

推荐用途

程序想要在写格式化（writing formatted ）json 时指定换行符。

默认开启

No

JSON_INDENT

此选项用来指定写格式化（writing formatted ）json 时指定缩进。默认 libjson 使用的是 ASCII tab 来进行缩进，你可以指定指定为其他缩进符，比如空格。

推荐用途

程序想要在写格式化（writing formatted ）json 时将 tabs 缩进改为空格。

默认开启

No

JSON_ESCAPE_WRITES

此选项用于写格式化（writing formatted ）json 时进行特殊字符的转换。此选项是默认开启的，否则遇到特殊字符比如换行，空格，tab 可能会导致 json 处理出问题。强烈建议开启此选项。

推荐用途

程序想要在写格式化（writing formatted ）json 时转换特殊字符。

默认开启

Yes

JSON_COMMENTS

此选项告诉 libjson 所要处理的 json 将含有注释或者可以进行注释。通过此选项将保留 json 注释否则将忽略。

推荐用途

程序中想要对 json 进行注释以便于人眼识别，这样使得 json 更加易懂和修改。

默认开启

Yes

JSON_WRITE_BASH_COMMENTS

此选项使得 libjson 使用 bash 的注释符来输出带注释的 json，可以使用#或者 C 方式的/**/来进行注释。

推荐用途

需要采用 bash 注释方式的应用，比如 doxygen 的文档。

默认开启

No

JSON_WRITE_SINGLE_LINE_COMMENTS

此选项使得 libjson 不能使用 C 方式的多行注释符来进行注释，可以采用类似于//来进行注释。

推荐用途

不能采用多行注释符的应用。

默认开启

No

JSON_VALIDATE

此选项提供了对 json 进行验证的功能接口。

推荐用途

程序中可能会收到无效 json。

要求

JSON_READ_PRIORITY 需要被开启，否则编译会失败。

默认开启

No

JSON_CASE_INSENSITIVE_FUNCTIONS

此选项提供了忽略大小写的功能接口，比如 at，get，find...

推荐用途

在程序中并不需要区分节点名的大小写。

默认开启

Yes

JSON_INDEX_TYPE

此选项可以改变用来统计子节点数目的数据类型。通常此类数据类型采用unsigned int，因为这样更加高效，但是在某些情况下却不合适。比如说在嵌入式系统下内存资源宝贵，你应该选择 short 甚至 char。又比如在64位系统下节点数目有可能超过 unsigned int 的范围。

推荐用途

比如在64位系统或者嵌入式系统下。

默认开启

No

JSON_BOOL_TYPE

此选项用来改变 bool 类型适应 C 接口的(早期标准 C 中不支持 bool 类型)。

推荐用途

基于 C 接口的应用。

默认开启

No

JSON_INT_TYPE

此选项使得 as_int 的数据类型为 int。如果不启用此选项，as_int 的数据类型为 long。

推荐用途

使得 as_int 的数据类型为 int，比如你想要提高精度。

默认开启

No

JSON_NUMBER_TYPE

此选项用来改变 as_float 的数据类型。如果此选项不开启，大多数情况下数据类型为 double，但是在 JSON_LESS_MEMORY 开启的情况下，被压缩成 float。

推荐用途

想要对 as_float 数据类型进行额外控制。

默认开启

No

JSON_STRING_HEADER

此选项可以改变 libjson 库的 string 类型，这样你可以你可以和其他库进行很好地结合使用。比如 QT 和 wxWigdets，string 会替换为他们自己的 string 类型。你必须使用 typedefed json_string 做定义。

推荐用途

使 libjson 和其他库结合使用。

默认开启

No

JSON_NO_EXCEPTIONS

此选项禁用了 libjson 接口的异常处理。

推荐用途

想要禁用 libjson 的异常处理。

默认开启

No

（注意：如果你开启了 JSON_PREPARSE 选项，异常处理将自动启用。）

JSON_DEPRECATED_FUNCTIONS

此选项使已经废弃的接口仍旧可以使用，但是编译器会给出相关警告。

推荐用途

为了兼容以前的版本。

默认开启

Yes

JSON_SECURITY_MAX_NEST_LEVEL

此选项被用来阻止 DoS（拒绝服务攻击）。黑客有时会使用深层嵌套的 json 来对服务进行攻击导致服务出现问题，此安全选项禁止使用过深的嵌套层次，LEVEL 必须被定义为整型。安全检查在 json 验证阶段进行，并不进行解析。

推荐用途

开放接口，用来抵御拒绝服务攻击

默认开启

Yes（128）

JSON_SECURITY_MAX_STRING_LENGTH

此选项被用来阻止 DoS（拒绝服务攻击）。黑客有时会使用过长 json 的字符

串来对服务进行攻击导致服务出现问题，此安全选项禁止使用过长的字符串，LENGTH 必须被定义为整型。安全检查在 json 验证阶段进行，并不进行解析。

推荐用途

开放接口，用来抵御拒绝服务攻击。

默认开启

Yes (33554432) - 32MB

JSON_SECURITY_MAX_STREAM_OBJECTS

此选项被用来阻止 DoS（拒绝服务攻击）。黑客有时会使用过长 json 的字符串和大量的小对象来对服务进行攻击导致服务出现问题，此安全选项禁止使用过长的字符串，LENGTH 必须被定义为整型。安全检查在 json 验证阶段进行，并不进行解析。

推荐用途

开放接口，用来抵御拒绝服务攻击。

默认开启

Yes (128)

JSON_UNIT_TEST

此选项用来对 libjson 进行单元测试，以此对 libjson 进行维护和调试。提供了主要接口的测试和一些测试案例。

推荐用途

维护阶段使用。

默认开启

No