Mini-XML 程序员开发手册, Version 2.5

目录

mxmlDelete
mxmlElementDeleteAttr
mxmlElementGetAttr
mxmlElementSetAttr
mxmlElementSetAttrf
mxmlEntityAddCallback
mxmlEntityGetName
mxmlEntityGetValue
mxmlEntityRemoveCallback
mxmlFindElement
mxmlIndexDelete
mxmlIndexEnum
mxmlIndexFind
mxmlIndexNew
mxmlIndexReset
mxmlLoadFd
mxmlLoadFile
mxmlLoadString
mxmlNewCDATA
mxmlNewCustom
mxmlNewElement
mxmlNewInteger
mxmlNewOpaque
mxmlNewReal
mxmlNewText
mxmlNewTextf
mxmlNewXML
mxmlRelease
mxmlRemove
mxmlRetain
mxmlSAXLoadFd
mxmlSAXLoadFile
mxmlSAXLoadString
mxmlSaveAllocString
mxmlSaveFd
mxmlSaveFile
mxmlSaveString
mxmlSetCDATA
mxmlSetCustom
mxmlSetCustomHandlers
mxmlSetElement
mxmlSetErrorCallback
mxmlSetInteger
mxmlSetOpaque

mxmlSetReal
mxmlSetText
mxmlSetTextf
mxmlSetWrapMargin
mxmlWalkNext
mxmlWalkPrev
类型定义(typedef)
mxml_attr_t
mxml_custom_destroy_cb_t
mxml_custom_load_cb_t
mxml_custom_save_cb_t
mxml_custom_t
mxml_element_t
mxml_error_cb_t
mxml_index_t
mxml_load_cb_t
mxml_node_t
mxml_save_cb_t
mxml_sax_cb_t
mxml_sax_event_t
mxml_text_t
mxml_value_t
结构(struct)
mxml_attr_s
mxml_custom_s
mxml_element_s
mxml_index_s
mxml_node_s
mxml_text_s
联合(union)
mxml_value_u
Constants
mxml_sax_event_e
mxml_type_e
XML 方案 (用于自动化文档生成工具 mxmldoc)

序言

这份程序员参考手册描述了 Mini-XML 2.5 版本, 一个小型的 XML 解析库，使用它可以使你的 C 或者 C++应用程序方便的进行 XML 数据文件的读写
Mini-XML 最初是为了 Gutenprint 项目而开发，目的是为了替换既大又笨重的 libxml2 库，想要实现一个小型且易于使用的一些东西. 它开始于 2003 年 6 月的一个早晨，当时罗伯特

发表了下面几句话到开发者列表：
"这真是糟糕，我们需要 libxml2,但反复看来，我们的 XML 解析器仅需要我们可以操作的一小部分。"
我做了以下回复:
"考虑到你使用 XML 仅在一个有限的范围中，那么只使用几百行代码来编写一个微型 XML (mini-XML) API，应该是很简单的。"
我接受了这个挑战，用了两天的时间进行疯狂的编码，并且公开发布了第一个 mini-XML 版本，总共是 696 行代码。然后，罗伯特迅速把 mini-XML 整合到 Gutenprint 中，并且移除了 libxml2 库
感谢很多不同的开发者给我的回馈和支持， 从那以后，Mini-XML 逐渐发展为一个提供更多完整的 XML 实现，当前已经高达 3441 行代码，但已经可以和 103893 行代码的 libxml2 2.6.9 版本相比较了。
译者：仅用了两天时间，作者真是大牛啊！我较喜欢 Mini-XML，我也用过 TinyXML，libexpat,libxml2 等解析器，相比之下 Mini-XML 实现了一个非常简洁且功能适用的解析器，很适合我的需求：DOM 型解析器、解析小型的 XML 文件，不进行错误恢复及校验，简单易用，且使用纯 ANSI-C 实现，方便移植到嵌入系统中。评价：很好很强大，而且很简单。Z.F
除了 Gutenprint ，mini-XML 当前已经应用于以下的项目/应用软件:
Common UNIX Printing System
CUPS Driver Development Kit
ZynAddSubFX
如果您希望将您的项目添加到此列表或者从此列表中删除，或者如果您有任何意见和想法，或者想要发布关于使用 mini-XML 的经验，请给我发电子邮件（ mxml@easysw.com ）
本文档组织结构
本手册由以下章节和附录组成:
第一章, "构建，安装，以及打包 Mini-XML ", 关于 mini-XML 在编译、安装以及打包方面的说明.
第二章, "Mini-XML 入门 ",如何在你的应用程序中使用 mini-XML.
第三章, "更多的 Mini-XML 编程技术 展示了使用 mini-XML 库的更多的方法。
第四章, "使用 mxmldoc 工具 ", 描述如何使用 mxmldoc(1) 程序来生成文档.
附录 A, "Mini-XML 许可信息 ",使用和发布 mini-XML 的条款及条件。
附录 B, "发行说明 ", 列出了每次 mini-XML 发布版本的改变信息.
附录 C, "库参考手册 ", 包含了关于 mini-XML 的完整参考信息,使用 mxmldoc 生成.
附录 D, "XML 方案 ", 显示了 mxmldoc 生成 XML 文件时使用的 XML 方案.
词法约定
在这篇手册中使用了一些字体和风格的约定.下面是一些例子含义和使用说明:
lpstat
lpstat(1)
命令名称;如果在一章中第一次提及这个系统命令或者函数，则后面跟随手册页编号。
译者：上面指 Linux 的 manpages 手册，使用 man 命令查看。Z.F

/var
/usr/share/cups/data/testprint.ps
文件或者目录.

Request ID is Printer-123
屏幕输出.

lp -d printer filename ENTER
用户输入的文字，特殊键如 ENTER 总是使用大写.

12.3
文本中的数字，使用(.)来表示小数点.

缩略词
下面是在本手册中使用的缩略词:
Gb
GB, 即 1073741824 字节，1*1024*1024*1024 字节

kb
即 1024 字节，1*1024 字节

Mb
兆, or 1048576 bytes,1*1024*1024 字节

UTF-8, UTF-16
统一的字符编码标准, 8-位 或 16-位

W3C
万维网联盟

XML
可扩展标记语言

其他参考
The Unicode Standard, Version 4.0, Addison-Wesley, ISBN 0-321-18578-1
定义了用于 XML 的 Unicode 字符集.

Extensible Markup Language (XML) 1.0 (Third Edition)
W3C 制定的 XML 标准.

法律资料

PURPOSE. See the GNU Library General Public License for more details.

第一章 构建，安装，以及打包 Mini-XML

这一章描述了如何在你的系统上使用源码包构建，安装以及打包 Mini-XML,你将需要一个 ANSI/ISO-C 兼容的 C 编译器来构建 Mini- XML,GCC 可以工作，这也是大多数厂家的 C 编译器。如果你需要在 Windows 平台上构建， 我们建议使用 Virtual C++环境和解决方案文件. 对于其他的操作系统，你在 C 编译器之外需要一个 POSIX 兼容的 shell 和 make 程序. .

编译 Mini-XML

Mini-XML 同时具备基于 autoconf 的配置脚本和 Virtual C++的解决方案，可以用来编译库和关联的其他工具.

使用 Visual C++ 编译

打开在目录 vcnet 下的 mxml.sln 解决方案. 选择需要构建的配置,"Debug" (缺省选项) 或者 "Release", 并且 从 Build 菜单中选择 Build Solution 项.

使用命令行工具编译

在你的系统上键入下面的命令来配置 Mini-XML 源代码:

```
./configure ENTER
```

缺省的安装前缀是 /usr/local , 这可以被覆盖使用 --prefix 选项:

```
./configure --prefix=/foo ENTER
```

其他配置选项可以使用命令--help 选项进行查看:

```
./configure --help ENTER
```

当你配置完毕,使用 make(1) 程序来构建并且运行测试程序来校验是否工作正常，如下：

```
make ENTER
```

安装 Mini-XML

如果你使用 Visual C++, 分别拷贝 mxml.lib 和 mxml.h 文件到 Visual C++ lib 和 include 目录.

否则,使用 make 命令和 install 参数来安装 Mini-XML 到配置的目录中:

```
make install ENTER
```

创建 Mini-XML 包

Mini-XML 包含两个文件可以被用来创建二进制发行包.第一个文件是 mxml.spec 可以被使用通过 rpmbuild(8) 软件来创建 Red Hat 包管理器("RPM")的发行包，通常用于 Linux 平台. rpmbuild 可以自己进行软件编译, 你可以为它提供一个 Mini-XML 的 tar 文件来编译整个包，使用以下命令:

```
rpmbuild -ta mxml-version
.tar.gz ENTER
```

第二个文件是 mxml.list 被用于 epm(1) 程序创建不同格式的软件包. epm 程序可以从以下网址获得:

```
http://www.easysw.com/epm/
```

使用 make 命令通过 epm 目标 来构建一个针对你的系统的便携的本地包:

```
make epm ENTER
```

为了你方便,这些包保存在子目录 dist 中.便携包利用脚本和 tar 文件安装软件到目标系统中. 在展开包文件后,使用 mxml.install 脚本来安装这个软件.

这些本地包可以是本地操作系统的原生格式:红帽 Linux 的 RPM , Debian Linux 的 DPKG, Solaris 的 PKG, 等等.使用相应的命令来安装这些原生包.

第二章 Mini-XML 入门

这一章描述了如何写一个程序使用 Mini-XML 来访问 XML 文件中的数据.Mini-XML 提供了以下功能:
在内存中创建和管理 XML 文档的函数.
读 UTF-8 和 UTF-16 编码的 XML 文件和字符串.
写 UTF-8 编码的 XML 文件和字符串.
支持任意的元素名称，属性以及属性值，没有任何其他限制，仅受限于有效内存.
支持整形、浮点、自定义("CDATA")和文本数据类型在"叶"节点.
提供"查找"、"索引"、以及"步进"函数可以很简单的访问 XML 文档中的数据.
Mini-XML 不进行基于"XML 方案(SCHEMA)"文件或者其他内容源定义信息的校验和其他类型的处理 ,也不支持其他组织所要求的 XML 规范.
基础知识
Mini-XML 提供的一个你需要包含的头文件:

```
#include <mxml.h>
```

把 Mini-XML 库连接到你的应用程序使用 -lmxml 选项:

```
gcc -o myprogram myprogram.c -lmxml ENTER
```

如果你已经安装 pkg-config(1) 软件, 你可以使用它来为你的安装确定适当的编译和连接选项:

```
pkg-config --cflags mxml ENTER
pkg-config --libs mxml ENTER
```

节点
每一块 XML 文件中的信息片断(元素、文本、数字)是一个存储在内存中的"节点(nodes)" . 节点使用 mxml_node_t 结构进行定义. 它的 type 成员变量定义了节点类型（element, integer, opaque, real, or text) 决定了需要从联合(union)类型的成员变量 value 中获取的值.
表 2-1: Mini-XML 节点值的成员变量 值 类型 节点成员
用户定义 void * node->value.custom.data
XML 元素 char * node->value.element.name
整数 int node->value.integer
不透明字符串 char * node->value.opaque
浮点数 double node->value.real
文本 char * node->value.text.string
译者：节点类型定义枚举参见：mxml_type_e 。 Mini-XML 中的节点类型定义和其他有些解析器有些不同，其中整数、浮点、和文本节点是指在一个 XML 元素中一系列的使用空格作为分割的值，每个元素可以拥有多个以上节点，并可以选择使用空格分开，如：<abc>aa bb cc</abc>，Mini-MXML 在使用参数：MXML_TEXT_CALLBACK 进行载入时，将在 abc 元素下面生成 3 个 text 类型的子节点。在创建时也可以使用同样的方式创建节点。整数和浮点也是同样方式，但如果转换失败则 MiniXML 报错。而不透明字符串类型(OPAQUE) 则不进行字符串分割，在载入时需要使用 MXML_OPAQUE_CALLBACK 参数，将所有字符串形成

一个子节点。详情见：使用加载回调函数 。 Z.F
每一个节点总是有一个成员变量:user_data 可以允许你为每一个节点关联你需要的应用数据.
新的节点可以使用以下函数进行创建 mxmlNewElement , mxmlNewInteger , mxmlNewOpaque , mxmlNewReal , mxmlNewText mxmlNewTextf mxmlNewXML . 只有 elements 可以拥有子节点，顶级节点必须是一个 element , 通常是<?xml version="1.0"?> 使用 mxmlNewXML() 函数创建的节点.
每个节点都有一些关联节点的指针，上(parent), 下(child), 左(prev), and 右(next) 相对应于当前节点. 如果你有一个 XML 文件如下所示:

```
<?xml version="1.0"?>
<data>
    <node>val1</node>
    <node>val2</node>
    <node>val3</node>
    <group>
        <node>val4</node>
        <node>val5</node>
        <node>val6</node>
    </group>
    <node>val7</node>
    <node>val8</node>
</data>
```

那么在内存中的文件节点树看上去如下所示:

```
?xml
  |
data
  |
node - node - node - group - node - node
  |      |      |      |        |      |
val1   val2   val3     |      val7   val8
                       |
                    node - node - node
                      |      |      |
                    val4   val5   val6
```

这里"-"指向下一个节点，"|"指向第一个子节点。
当你使用完毕这些 XML 数据后，使用函数 mxmlDelete 来释放指定节点或者整个 XML 树节点和它下面所有子节点的内存:

```
mxmlDelete(tree);
```

创建 XML 文档
你可以在内存中创建和更新 XML 文档，使用 mxmlNew 一系列函数. 下面的代码将创建上一章描述的 XML 文档:

```
mxml_node_t *xml;
mxml_node_t *data;
mxml_node_t *node;
```

```
mxml_node_t *group;
xml = mxmlNewXML("1.0");
data = mxmlNewElement(xml, "data");
    node = mxmlNewElement(data, "node");
    mxmlNewText(node, 0, "val1");
    node = mxmlNewElement(data, "node");
    mxmlNewText(node, 0, "val2");
    node = mxmlNewElement(data, "node");
    mxmlNewText(node, 0, "val3");
    group = mxmlNewElement(data, "group");
        node = mxmlNewElement(group, "node");
        mxmlNewText(node, 0, "val4");
        node = mxmlNewElement(group, "node");
        mxmlNewText(node, 0, "val5");
        node = mxmlNewElement(group, "node");
        mxmlNewText(node, 0, "val6");
    node = mxmlNewElement(data, "node");
    mxmlNewText(node, 0, "val7");
    node = mxmlNewElement(data, "node");
    mxmlNewText(node, 0, "val8");
```

我们首先使用 mxmlNewXML 函数来创建所有 XML 文件都需要的标头 <?xml version="1.0"?> :

```
xml = mxmlNewXML("1.0");
```

然后我们使用 mxmlNewElement 函数来创建本文件使用的<data> 节点.第一个参数指定了父节点(xml) ，第二个参数是元素名 (data):

```
data = mxmlNewElement(xml, "data");
```

每个在本文件中<node>...</node> 之间的部分使用函数 mxmlNewElement 和 mxmlNewText 来创建. mxmlNewText 的第一个参数指定了父节点 (node).第二个参数指定了是否在文本之前添加空白字符，在本例中使用 0 或者 false.最后一个参数指定了需要添加的实际文本:

```
node = mxmlNewElement(data, "node");
mxmlNewText(node, 0, "val1");
```

在内存中的 XML 结果可以被保存或者进行其他处理，就像一个从磁盘或者字符串中读取的文档一样.

加载 XML

你可以加载一个 XML 文件使用函数 mxmlLoadFile :

```
FILE *fp;
mxml_node_t *tree;
fp = fopen("filename.xml", "r");
tree = mxmlLoadFile(NULL, fp,
                    MXML_TEXT_CALLBACK);
fclose(fp);
```

第一个参数如果有的话则指定了一个存在的 XML 父节点.一般你将这个参数等于 NULL ,除非你想要连接多个 XML 源. 如果此参数等于 NULL ，那么指定的 XML 文件必须是一个完

整的 XML 文档，文档头部要包含?xml 元素.
第二个参数指定了一个标准的文件流，使用 fopen() 或者 popen() 进行打开. 你也可以使用 stdin ，如果你想要实现一个 XML 过滤器程序.
第三个参数指定了一个回调函数用于一个新的 XML 元素节点直接返回的子节点的值类型: MXML_CUSTOM , MXML_IGNORE , MXML_INTEGER , MXML_OPAQUE , MXML_REAL , or MXML_TEXT . 加载回调函数的细节在第三章 做了详细描述. 示例代码使用 MXML_TEXT_CALLBACK 常量指定文档中所有的数据节点都包含使用以空格字符分割的文本的值. 其他标准的回调还有 MXML_IGNORE_CALLBACK , MXML_INTEGER_CALLBACK , MXML_OPAQUE_CALLBACK , 和 MXML_REAL_CALLBACK .
函数 mxmlLoadString 可以从一个字符串中载入 XML 节点树:

```
char buffer[8192];
mxml_node_t *tree;
...
tree = mxmlLoadString(NULL, buffer,
                      MXML_TEXT_CALLBACK);
```

第一个和第三个参数和 mxmlLoadFile() 用法一样. 第二个参数指定了指定了字符串或者字符缓冲区用于加载XML，当父节点参数为NULL 时内容必须为完整的XML 文档，包括XML 头?xml 元素.
保存 XML
你可以保存 XML 文件使用 mxmlSaveFile 函数:

```
FILE *fp;
mxml_node_t *tree;
fp = fopen("filename.xml", "w");
mxmlSaveFile(tree, fp, MXML_NO_CALLBACK);
fclose(fp);
```

第一个参数为想要保存的 XML 节点树,一般应该是一个指向你的 XML 文档顶级节点?xml 的节点指针.
第二个单数是一个标准文件流，使用 fopen() 或者 popen() 来打开. 你也可以使用 stdout 如果你想要实现一个 XML 过滤器程序.
第三个参数是一个空白回调函数用来控制保存文件时插入的"空白"字符."空白回调"的详细信息参见 第三章 . 以上的示例代码使用了 MXML_NO_CALLBACK 常量来指定不需要特别的空白处理.
函数 mxmlSaveAllocString , 和 mxmlSaveString 保存 XML 节点树到一个字符串中:

```
char buffer[8192];
char *ptr;
mxml_node_t *tree;
...
mxmlSaveString(tree, buffer, sizeof(buffer),
               MXML_NO_CALLBACK);
...
ptr = mxmlSaveAllocString(tree, MXML_NO_CALLBACK);
```

第一个和最后一个参数的用法和函数 mxmlSaveFile() 一样. 函数 mxmlSaveString 给出了一个指针和长度的参数来保存 XML 文档到一个固定大小的缓冲区中。，

mxmlSaveAllocString() 返回了使用 malloc 分配的一个字符串缓冲区 malloc() .

自动折行控制

当我们保存 XML 文档时, Mini-XML 一般在第 75 列进行折行，因为这样在终端下最易读. 函数 mxmlSetWrapMargin 可以覆盖缺省的折行界限:

```
mxmlSetWrapMargin(132);

mxmlSetWrapMargin(0);
```

搜索和遍历节点

函数 mxmlWalkPrev and mxmlWalkNext 可以被用来遍历 XML 节点树:

```
mxml_node_t *node;

node = mxmlWalkPrev(current, tree,
                    MXML_DESCEND);
node = mxmlWalkNext(current, tree,
                    MXML_DESCEND);
```

另外,你可以搜索一个命名的 XML 元素/节点,使用函数 mxmlFindElement :

```
mxml_node_t *node;

node = mxmlFindElement(tree, tree, "name",
                       "attr", "value",
                       MXML_DESCEND);
```

参数 name , attr , 和 value 可以被设置为 NULL 作为全部匹配, e.g.:

```
node = mxmlFindElement(tree, tree, "a",
                       NULL, NULL,
                       MXML_DESCEND);

node = mxmlFindElement(tree, tree, "a",
                       "href", NULL,
                       MXML_DESCEND);

node = mxmlFindElement(tree, tree, "a",
                       "href",
                       "http://www.easysw.com/",
                       MXML_DESCEND);

node = mxmlFindElement(tree, tree, NULL,
                       "src", NULL,
                       MXML_DESCEND);

node = mxmlFindElement(tree, tree, NULL,
                       "src", "foo.jpg",
                       MXML_DESCEND);
```

你也可以使用同样的功能进行遍历:

```
mxml_node_t *node;
for (node = mxmlFindElement(tree, tree,
                            "name",
                            NULL, NULL,
                            MXML_DESCEND);
     node != NULL;
     node = mxmlFindElement(node, tree,
                            "name",
                            NULL, NULL,
                            MXML_DESCEND))
{
  ... do something ...
}
```

参数 MXML_DESCEND 可以是下面三个常量之一:

MXML_NO_DESCEND 含义是不查看任何的子节点在 XML 元素层次中，仅查看同级的伙伴节点或者父节点直到到达顶级节点或者给出的树的顶级节点.

"group"节点的上一个节点时它左面的"node"子节点,下一个节点是"group"右面的"node"子节点..

MXML_DESCEND_FIRST 含义是向下搜索到一个节点的第一个匹配子节点，但不再继续向下搜索。你一般使用于遍历一个父节点的直接的子节点。 ,如：在上面的例子中的所有在"?xml"父节点下的所有的"node"和"group"子节点。.

这个模式仅适用于搜索(search)功能，遍历功能(walk)对待它和 MXML_DESCEND 一样，因为每次调用都是首次调用。

MXML_DESCEND 含义是一直向下直到树的根部. "group"节点的上一个节点将是"val3"节点，下一个节点将是"group"的下面的第一个子节点。

如果你要使用函数 mxmlWalkNext() 从根结点"?xml" 遍历到整个树的结束, 那么这个顺序将如下所示：

?xml data node val1 node val2 node val3 group node val4 node val5 node val6 node val7 node val8

如果你从"val8"开始并使用函数 mxmlWalkPrev() 进行遍历, 这个顺序将是反的，结束于"?xml"节点.

第三章 更多的 Mini-XML 编程技术

这一章显示了更多的在你的应用程序中使用 Mini-XML 的方法。

Load Callbacks

第二章 介绍了函数 mxmlLoadFile() 和 mxmlLoadString() . 这些函数的最后一个参数是一个回调函数，决定了在一个 XML 文档中每个数据节点的值的类型。

Mini-XML 为简单 XML 数据文件定义了几个标准的回调函数：

MXML_INTEGER_CALLBACK - 所有的数据节点包含以空格分割的整数。

MXML_OPAQUE_CALLBACK - 所有的数据节点包含"不透明"字符串（CDATA）。

MXML_REAL_CALLBACK - 所有的数据节点包含以空格分割的浮点数。
MXML_TEXT_CALLBACK - 所有的数据节点包含以空格分割的文本字符串。
你可以为更复杂的 XML 文档提供你自己的回调函数。你的回调函数将接收到一个到当前 XML 元素节点的指针并且必须为这个 XML 元素节点返回一个直接子节点的值类型: MXML_INTEGER , MXML_OPAQUE , MXML_REAL , 或 MXML_TEXT .这个函数在这个 XML 元素和它的全部属性被读取以后 被调用，所以你可以查看这个 XML 元素的名称、属性以及属性的值来决定适当的返回值类型。
下面的回调函数查看一个名称为"type"的属性或者 XML 元素的名字来决定它的子节点的值类型：

```
mxml_type_t
type_cb(mxml_node_t *node)
{
  const char *type;

  type = mxmlElementGetAttr(node, "type");
  if (type == NULL)
type = node->value.element.name;
  if (!strcmp(type, "integer"))
return (MXML_INTEGER);
  else if (!strcmp(type, "opaque"))
return (MXML_OPAQUE);
  else if (!strcmp(type, "real"))
return (MXML_REAL);
  else
return (MXML_TEXT);
}
```

要使用这个回调函数，只需要在你调用任何加载函数时简单的使用它的名字：

```
FILE *fp;
mxml_node_t *tree;
fp = fopen("filename.xml", "r");
tree = mxmlLoadFile(NULL, fp, type_cb
);
fclose(fp);
```

保存回调
第二章 也介绍了 mxmlSaveFile() , mxmlSaveString() , 和 mxmlSaveAllocString() 函数。这些函数的最后一个参数是一个回调函数被用来自动在一个 XML 文档中添加空白字符。
你的回调函数将在每个 XML 元素被调用四次，传入参数为一个到这个节点的指针和一个 "where" 的值： MXML_WS_BEFORE_OPEN , MXML_WS_AFTER_OPEN , MXML_WS_BEFORE_CLOSE , 或者 MXML_WS_AFTER_CLOSE 。如果不需要插入空白字符这个回调函数将返回 NULL ，否则返回字符串（空白、跳格、回车和换行）将被插入。
下面的空白回调可以被用来为 XHTML 输出添加空白字符，来使它在一般的文本编辑器中更加易读：

```
const char *
whitespace_cb(mxml_node_t *node,
```

```
                        int where)
    {
      const char *name;

      name = node->value.element.name;
      if (!strcmp(name, "html") ||
          !strcmp(name, "head") ||
          !strcmp(name, "body") ||
  !strcmp(name, "pre") ||
          !strcmp(name, "p") ||
  !strcmp(name, "h1") ||
          !strcmp(name, "h2") ||
          !strcmp(name, "h3") ||
  !strcmp(name, "h4") ||
          !strcmp(name, "h5") ||
          !strcmp(name, "h6"))
      {

 if (where == MXML_WS_BEFORE_OPEN ||
             where == MXML_WS_AFTER_CLOSE)
   return ("\n");
      }
      else if (!strcmp(name, "dl") ||
                  !strcmp(name, "ol") ||
                  !strcmp(name, "ul"))
      {

 return ("\n");
      }
      else if (!strcmp(name, "dd") ||
                  !strcmp(name, "dt") ||
                  !strcmp(name, "li"))
      {

 if (where == MXML_WS_BEFORE_OPEN)
   return ("\t");
 else if (where == MXML_WS_AFTER_CLOSE)
   return ("\n");
      }

      return (NULL);
    }
```

要使用这些回调函数，只需要在你调用任何保存函数时简单使用它的名字：

```
    FILE *fp;
```

```
    mxml_node_t *tree;
    fp = fopen("filename.xml", "w");
    mxmlSaveFile(tree, fp, whitespace_cb
);
    fclose(fp);
```

用户定义数据类型

Mini-XML 支持通过全局的载入和保存回调函数使用自定义数据类型。 每次只能有一组回调函数被同时激活，然而你的回调函数可以为支持多种所需要的自定义数据类型来保存更多的信息。 节点类型 MXML_CUSTOM 表示一个自定义数据内容的节点。

加载回调接收一个到当前数据节点的指针和一个不透明字符串数据从 XML 源中并且将字符集转换为 UTF-8 编码。例如：如果我们想要支持定制的日期/时间类型并且编码为 "yyyy-mm-ddThh:mm:ssZ" (ISO 格式), 那么加载回调函数如下所示：

```
    typedef struct
    {
      unsigned          year,
                        month,
                        day,
                        hour,
                        minute,
                        second;
      time_t            unix;
    } iso_date_time_t;
    int
    load_custom(mxml_node_t *node,
                const char *data)
    {
      iso_date_time_t *dt;
      struct tm tmdata;

      dt = calloc(1, sizeof(iso_date_time_t));

      if (sscanf(data, "%u-%u-%uT%u:%u:%uZ",
                 &(dt->year), &(dt->month),
                 &(dt->day), &(dt->hour),
                 &(dt->minute),
                 &(dt->second)) != 6)
      {

        free(dt);
        return (-1);
      }

      if (dt->month <1 || dt->month > 12 ||
          dt->day   <1 || dt->day > 31 ||
```

```
            dt->hour   <0 || dt->hour > 23 ||
            dt->minute   <0 || dt->minute > 59 ||
            dt->second   <0 || dt->second > 59)
        {

          free(dt);
          return (-1);
        }

        tmdata.tm_year = dt->year - 1900;
        tmdata.tm_mon    = dt->month - 1;
        tmdata.tm_day    = dt->day;
        tmdata.tm_hour = dt->hour;
        tmdata.tm_min    = dt->minute;
        tmdata.tm_sec    = dt->second;
        dt->unix = gmtime(&tmdata);

        node->value.custom.data       = dt;
        node->value.custom.destroy = free;

        return (0);
      }
```

这个函数成功时返回 0，当不能正确解码自定义数据或者数据内容错误时返回-1。自定义数据节点包含一个 void 指针用来保存这个节点已经分配的自定义数据，还有一个指向销毁函数的指针用于当节点被删除时释放自定义数据。

保存回调接收一个节点指针并且返回一个包含自定义数据值的已经分配的字符串。下面的保存回调函数可以被用来保存我们的 ISO 日期/时间类型：

```
      char *
      save_custom(mxml_node_t *node)
      {
        char data[255];
        iso_date_time_t *dt;

        dt = (iso_date_time_t *)node->custom.data;
        snprintf(data, sizeof(data),
                 "u-u-uTu:u:uZ",
                 dt->year, dt->month, dt->day,
                 dt->hour, dt->minute, dt->second);
        return (strdup(data));
      }
```

你可以注册这些回调函数使用 mxmlSetCustomHandlers() 函数：

```
      mxmlSetCustomHandlers(load_custom
,save_custom
);
```

改变节点的值

到现在为止所有的例子集中描述了如何创建和加载新的 XML 数据节点。然而，许多的应用程序在它们的工作中需要操纵或者改变节点，所以 Mini-XML 提供了一些函数来安全的改变节点的值并且不会发生内存泄漏。

已有的节点可以被改变通过使用函数 mxmlSetElement() , mxmlSetInteger() , mxmlSetOpaque() , mxmlSetReal() , mxmlSetText() , 和 mxmlSetTextf() 。例如：使用下面的函数调用可以改变一个文本节点到包含字符串"new"并且具有前导的空白字符：

```
mxml_node_t *node;
mxmlSetText(node, 1, "new");
```

格式化的文本

mxmlNewTextf() 和 mxmlSetTextf() 函数分别是创建和改变文本节点，使用 printf -风格的格式字符串和参数。例如：使用下面的函数调用来创建一个新的文本节点包含一个构造的文件名：

```
mxml_node_t *node;

node = mxmlNewTextf(node, 1, "%s/%s",
                    path, filename);
```

索引

Mini-XML 提供了一些函数来管理节点的索引。当前的实现提供了同样的功能就像 mxmlFindElement() 一样。使用索引优势是可以使搜索和枚举 XML 元素显著加快。唯一不利的是每个索引都是一个关于这个 XML 文档的静态快照，所以索引不适合当相对于它的搜索操作，XML 数据更加频繁更新时的情况。上面的创建一个索引近似相当于遍历这个 XML 文档树。在索引中的节点被按照 XML 元素名和它的参数值进行排序。

这些索引被保存在 mxml_index_t 结构中。用 mxmlIndexNew() 函数可以创建一个新的索引：

```
mxml_node_t *tree;
mxml_index_t *ind;
ind = mxmlIndexNew(tree, "element",
                   "attribute");
```

第一个参数是需要进行索引的 XML 节点树。通常是一个指向?xml 元素的节点。

第二个参数包含了需要进行索引的 XML 元素；使用 NULL 值将按照字母顺序索引所有的 XML 元素节点。

第三个参数包含了需要进行索引的属性；使用 NULL 将使只有 XML 元素名字被索引。

当索引被建立后，函数 mxmlIndexEnum() , mxmlIndexFind() , and mxmlIndexReset() 可以被用来访问索引中的节点。函数 mxmlIndexReset() 被用来重置在索引中的"当前"节点指针，允许你在同一个索引中进行新的搜索和枚举遍历。典型应用是你将在你调用函数 mxmlIndexEnum() 和 mxmlIndexFind() 之前调用这个函数。

函数 mxmlIndexEnum() 用来枚举在索引中的每一个节点，可以在一个循环中使用，如下所示：

```
mxml_node_t *node;
mxmlIndexReset(ind);
while ((node = mxmlIndexEnum(ind)) != NULL)
{
  // do something with node
}
```

函数 mxmlIndexFind() 定位下一次在索引中出现的 XML 元素名和属性值。它可以被用于发现在索引中的所有匹配的 XML 元素，如下所示：

```
mxml_node_t *node;
mxmlIndexReset(ind);
while ((node = mxmlIndexFind(ind, "element",
                             "attr-value"))
       != NULL)
{
  // do something with node
}
```

第二和第三个参数分别表示 XML 元素名和属性值。使用 NULL 指针用来返回索引中所有的 XML 元素或者属性。如果 XML 元素名和属性值同时为 NULL 则相当于调用函数 mxmlIndexEnum 。

当我们使用完这个索引后，使用函数 mxmlIndexDelete() 来删除它:

```
mxmlIndexDelete(ind);
```

SAX (流方式解析) 加载文档

Mini-XML 支持一个关于简单 XML API (SAX)的实现，以允许你通过节点流的方式加载和处理 XML 文档。另外允许你处理任何大小的 XML 文件，Mini-XML 的实现也允许你为下一步的处理而只在内存中保留 XML 文档的一部分。

The mxmlSAXLoadFd , mxmlSAXLoadFile , 和 mxmlSAXLoadString 函数提供了 SAX 加载的 API。每个函数工作起来就象 mxmlLoad 函数一样，但是使用一个回调函数来处理它读到的每一个节点。

回调函数接收到一个节点，一个事件代码和一个你提供的用户数据指针：

```
void
sax_cb(mxml_node_t *node,
       mxml_sax_event_t event,
       void *data)
{
  ... do something ...
}
```

事件(event)将是下面的其中一个：

MXML_SAX_CDATA - CDATA 正在被读取

MXML_SAX_COMMENT - 一个注释正在被读取

MXML_SAX_DATA - 数据(custom, integer, opaque, real, or text) 正在被读取

MXML_SAX_DIRECTIVE - 一个处理指令正在被读取

MXML_SAX_ELEMENT_OPEN - 一个"打开"元素节点正在被读取，如(<element>)

MXML_SAX_ELEMENT_CLOSE - 一个"关闭"元素节点正在被读取，如(</element>)

XML 元素将被 释放 在一个"关闭"元素被处理后。所有其他的节点在他们被处理后都被释放。SAX 回调函数可以 保留 这个节点，通过调用函数 mxmlRetain 。例如：下面的 SAX 回调函数将保留所有的节点，就像一个普通的内存加载一样：

```
void
sax_cb(mxml_node_t *node,
       mxml_sax_event_t event,
       void *data)
```

```
{
  if (event != MXML_SAX_ELEMENT_CLOSE)
    mxmlRetain(node);
}
```

更多的典型 SAX 回调函数将只保留后面需要处理的这个 XML 文档的一小部分。例如，下面的 SAX 回调函数将保留一个 XHTML 文件的标题和头部信息。它总是保留一个 XML 元素（父节点）就像<html> , <head> , 和 <body> ,并处理指令节点就像 <?xml ... ?> 和 <!DOCTYPE ... > ：

```
void
sax_cb(mxml_node_t *node,
       mxml_sax_event_t event,
       void *data)
{
  if (event == MXML_SAX_ELEMENT_OPEN)
  {

    char *name = node->value.element.name;
    if (!strcmp(name, "html") ||
        !strcmp(name, "head") ||
        !strcmp(name, "title") ||
        !strcmp(name, "body") ||
        !strcmp(name, "h1") ||
        !strcmp(name, "h2") ||
        !strcmp(name, "h3") ||
        !strcmp(name, "h4") ||
        !strcmp(name, "h5") ||
        !strcmp(name, "h6"))
      mxmlRetain(node);
  }
  else if (event == MXML_SAX_DIRECTIVE)
    mxmlRetain(node);
  else if (event == MXML_SAX_DATA &&
           node->parent->ref_count > 1)
  {

    mxmlRetain(node);
  }
}
```

这个结果框架文档树可以被搜素就像一个使用函数 mxmlLoad 加载的一样。例如，一个过滤器程序用来从标准输入(stdin)中读取一个 XHTML 文档，并显示在文档中的标题和标头，如下所示：

```
mxml_node_t *doc, *title, *body, *heading;
doc = mxmlSAXLoadFd(NULL, 0,
                    MXML_TEXT_CALLBACK,
```

```
                                    sax_cb
, NULL);
    title = mxmlFindElement(doc, doc, "title",
                                      NULL, NULL,
                                      MXML_DESCEND);
    if (title)
      print_children(title);
    body = mxmlFindElement(doc, doc, "body",
                                     NULL, NULL,
                                     MXML_DESCEND);
    if (body)
    {
      for (heading = body->child;
           heading;
           heading = heading->next)
        print_children(heading);
    }
```

第四章 使用 mxmldoc 工具

这一章描述了如何使用 mxmldoc(1) 程序自动从 c 和 c++源文件中生成文档。

基础知识

最初开发是用来生成 Mini-XML 和 CUPS API 的文档，现在 mxmldoc 是一个通用工具实现了扫描 C 和 C++源文件以生成 HTML 和 man 手册页文档与一个描述这些源文件中的函数、类型和宏的定义的 XML 文件。不像一些流行的文档生成工具如 Doxygen 或 Javadoc, mxmldoc 使用"在线"注释并不是注释头文件，允许更加自然的代码文档。

缺省情况下，mxmldoc 生成 HTML 文档个，例如：下面的命令将扫描所有的在当前目录下的 C 源代码和头文件并生成一个 HTML 文档文件叫 filename.html :

```
    mxmldoc *.h *.c >filename.html ENTER
```

你也可以指定创建一个包含所有源文件信息的 XML 文件。例如，下面的命令创建一个 XML 文件 filename.xml 并添加到这个 HTML 文件中：

```
    mxmldoc filename.xml *.h *.c >filename.html ENTER
```

--no-output 选项关闭标准的 HTML 输出：

```
    mxmldoc --no-output filename.xml *.h *.c ENTER
```

你可以再次运行 mxmldoc 通过这个 XML 文件来生成 HTML 文档：

```
    mxmldoc filename.xml >filename.html ENTER
```

--man filename 选项来告诉 mxmldoc 创建一个手册页(man page)代替 HTML 文档，例如：

```
    mxmldoc --man filename filename.xml \
        >filename.man ENTER

    mxmldoc --man filename *.h *.c \
        >filename.man ENTER
```

```
mxmldoc --man filename filename.xml *.h *.c \
    >filename.man ENTER
```

注释你的代码

在上面已经提到，mxmldoc 查看在线注释来描述你源文件中的函数、类型以及常量。Mxmldoc 将为你源文件中"所有"公开的命名生成文档，所有以下划线开始的名称和被使用@private@ 指令注解的名称被认为是“私有”的而不生成文档。

出现在函数或者类型定义前面的注释被用来生出这个函数或者类型的文档。出现在参数、定义、返回类型或者变量定义被用来生成它们的注释。例如：下列代码片断定义一个包含 key/vaue 的结构以及一个创建这个结构的新实例的函数：

```
struct keyval
{
  char *key;
  char *val;
};

struct keyval *
new_keyval(
    const char *key,
 const char *val)
{
  ...
}
```

Mxmldoc 总是知道并从注释字符串中移除多余的(*)号，所以下面的这个注释字符串：

将被显示为：

```
Compute the value of PI.
The function connects to an Internet server
that streams audio of mathematical monks
chanting the first 100 digits of PI.
```

注释 中也可以包含@name ...@ 这样的特殊指令字符串：

@deprecated@ - 标识一个条目为不支持并阻止它的使用

@private@ - 标识一个条目为为私有的所以它将不被包含在文档中。

@since ...@ - 标识一个条目为为从一个特别重要的发布更新。从@since 到@ 间的文本在文档中将被高亮显示，如： @since CUPS 1.3@ 。

标题、分段名和简介

Mxmldoc 也提供了一些选项来设置生成的文档中的标题、分段名和简介。The --title text 选项指定了文档的标题。标题字符串通常放在引号内：

```
mxmldoc filename.xml \
    --title "My Famous Documentation" \
    >filename.html ENTER
```

--section name 选项指定了文档的分段名。对于 HTML 文档，这个名字放在一个 HTML 注释块中，如下所示：

```
<!-- SECTION: name -->
```

对于手册页(man pages), 分段名通常是一个数字("3")，或者是一个数字后面跟着厂商名如("3acme")。这个分段名被使用在手册页(man pages)中的.TH 指令里：

.TH mylibrary 3acme "My Title" ...

缺省手册页(man pages)的分段名输出是"3"。对于 HTML 输出没有缺省的分段名。

最后，--intro filename 选项指定了一个文件来嵌入到在标题和分段名之后和生成文档的前面。对于 HTML 文档，这个文件必须兼容 HTML 并且不能使用 DOCTYPE , html , and body 元素。 .对于手册页文档，这个文件必须兼容有效的 nroff(1) 文本.

附录 A Mini-XML 许可信息

The Mini-XML library and included programs are provided under the terms of the GNU Library General Public License (LGPL) with the following exceptions:

LGPL 许可协议中文版本参见：GNU 通用公共许可证 Z.F

Static linking of applications to the Mini-XML library does not constitute a derivative work and does not require the author to provide source code for the application, use the shared Mini-XML libraries, or link their applications against a user-supplied version of Mini-XML.

If you link the application to a modified version of Mini-XML, then the changes to Mini-XML must be provided under the terms of the LGPL in sections 1, 2, and 4.

You do not have to provide a copy of the Mini-XML license with programs that are linked to the Mini-XML library, nor do you have to identify the Mini-XML license in your program or documentation as required by section 6 of the LGPL.

GNU LIBRARY GENERAL PUBLIC LICENSE

Version 2, June 1991

Copyright (C) 1991 Free Software Foundation, Inc.

59 Temple Place - Suite 330, Boston, MA 02111-1307, USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

[This is the first released version of the library GPL. It is numbered 2 because it goes with version 2 of the ordinary GPL.]

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users.

This license, the Library General Public License, applies to some specially designated Free Software Foundation software, and to any other libraries whose authors decide to use it. You can use it for your libraries, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or

to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library, or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link a program with the library, you must provide complete object files to the recipients so that they can relink them with the library, after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

Our method of protecting your rights has two steps: (1) copyright the library, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the library.

Also, for each distributor's protection, we want to make certain that everyone understands that there is no warranty for this free library. If the library is modified by someone else and passed on, we want its recipients to know that what they have is not the original version, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that companies distributing free software will individually obtain patent licenses, thus in effect transforming the program into proprietary software. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License, which was designed for utility programs. This license, the GNU Library General Public License, applies to certain designated libraries. This license is quite different from the ordinary one; be sure to read it in full, and don't assume that anything in it is the same as in the ordinary license.

The reason we have a separate public license for some libraries is that they blur the distinction we usually make between modifying or adding to a program and simply using it. Linking a program with a library, without changing the library, is in some sense simply using the library, and is analogous to running a utility program or application program. However, in a textual and legal sense, the linked executable is a combined work, a derivative of the original library, and the ordinary General Public License treats it as such.

Because of this blurred distinction, using the ordinary General Public License for libraries did not effectively promote software sharing, because most developers did not use the libraries. We concluded that weaker conditions might promote sharing better.

However, unrestricted linking of non-free programs would deprive the users of those programs of all benefit from the free status of the libraries themselves. This Library General Public License is intended to permit developers of non-free programs to use free libraries, while preserving your freedom as a user of such programs to change the free libraries that are incorporated in them. (We have not seen how to achieve this as regards changes in header files, but we have achieved it as regards changes in the actual functions of the Library.) The hope is that this will lead to faster development of free libraries.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the libary" and a "work that uses the library". The former contains code derived from the library, while the latter only works together with the library.

Note that it is possible for a library to be covered by the ordinary General Public License rather than by this special one.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Library General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) The modified work must itself be a software library.

b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application

does not supply it, the square root function must still compute square roots.)
These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.
Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.
In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.
3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.
Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.
This option is useful when you wish to copy part of the code of the Library into a program that is not a library.
4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.
If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.
5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.
However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.
When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is

not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also compile or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

c) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

d) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of

the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or

by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Library General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.
one line to give the library's name and an idea of what it does.
Copyright (C) year name of author
This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.
This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.
You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
Also add information on how to contact you by electronic and paper mail.
You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the library, if necessary. Here is a sample; alter the names:
Yoyodyne, Inc., hereby disclaims all copyright interest in the library `Frob' (a library for tweaking knobs) written by James Random Hacker.
signature of Ty Coon , 1 April 1990 Ty Coon, President of Vice
That's all there is to it!

附录 B 发行说明

Changes in Mini-XML 2.5
The mxmldoc program now makes greater use of CSS and supports a --css option to embed an alternate stylesheet.
The mxmldoc program now supports --header and --footer options to insert documentation content before and after the generated content.
The mxmldoc program now supports a --framed option to generate framed HTML output.
The mxmldoc program now creates a table of contents including any headings in the --intro file when generating HTML output.
The man pages and man page output from mxmldoc did not use "\-" for dashes (STR #68)
The debug version of the Mini-XML DLL could not be built (STR #65)
Processing instructions and directives did not work when not at the top level of a document (STR #67)
Spaces around the "=" in attributes were not supported (STR #67)
Changes in Mini-XML 2.4
Fixed shared library build problems on HP-UX and Mac OS X.
The mxmldoc program did not output argument descriptions for functions properly.
All global settings (custom, error, and entity callbacks and the wrap margin) are now managed separately for each thread.

Added mxmlElementDeleteAttr() function (STR #59)
mxmlElementSetAttrf() did not work (STR #57)
mxmlLoad*() incorrectly treated declarations as parent elements (STR #56)
mxmlLoad*() incorrectly allowed attributes without values (STR #47)
Fixed Visual C++ build problems (STR #49)
mxmlLoad*() did not return NULL when an element contained an error (STR #46)
Added support for the apos character entity (STR #54)
Fixed whitespace detection with Unicode characters (STR #48)
mxmlWalkNext() and mxmlWalkPrev() did not work correctly when called with a node with no children as the top node (STR #53)
Changes in Mini-XML 2.3
Added two exceptions to the LGPL to support static linking of applications against Mini-XML
The mxmldoc utility can now generate man pages, too.
Added a mxmlNewXML() function
Added a mxmlElementSetAttrf() function (STR #43)
Added a snprintf() emulation function for the test program (STR #32)
Added the _CRT_SECURE_NO_DEPRECATE definition when building on VC++ 2005 (STR #36)
mxmlLoad*() did not detect missing > characters in elements (STR #41)
mxmlLoad*() did not detect missing close tags at the end of an XML document (STR #45)
Added user_data and ref_count members to mxml_node_t structure
Added mxmlReleaseNode() and mxmlRetainNode() APIs for reference-counted nodes
Added mxmlSetWrapMargin() to control the wrapping of XML output
Added conditional check for EINTR error code for certain Windows compilers that do not define it (STR #33)
The mxmldoc program now generates correct HTML 4.0 output - previously it generated invalid XHTML
The mxmldoc program now supports "@deprecated@, "@private@", and "@since version@" comments
Fixed function and enumeration type bugs in mxmldoc
Fixed the XML schema for mxmldoc
The mxmldoc program now supports --intro, --section, and --title options
The mxmlLoad*() functions could leak a node on an error (STR #27)
The mxml_vsnprintf() function could get in an infinite loop on a buffer overflow (STR #25)
Added new mxmlNewCDATA() and mxmlSetCDATA() functions to create and set CDATA nodes, which are really just special element nodes
Added new MXML_IGNORE type and MXML_IGNORE_CB callback to ignore non-element nodes, e.g. whitespace
mxmlLoad*() did not treat custom data as opaque, so whitespace characters would be lost
Changes in Mini-XML 2.2.2
mxmlLoad*() did not treat custom data as opaque, so whitespace characters would be lost.
Changes in Mini-XML 2.2.1
mxmlLoadFd(), mxmlLoadFile(), and mxmlLoadString() now correctly return NULL on error (STR #21)

mxmlNewInteger(), mxmlNewOpaque(), mxmlNewReal(), mxmlNewText(), and mxmlNewTextf() incorrectly required a parent node (STR #22)
Fixed an XML output bug in mxmldoc.
The "make install" target now uses the install command to set the proper permissions on UNIX/Linux/OSX.
Fixed a MingW/Cygwin compilation problem (STR #18)
Changes in Mini-XML 2.2
Added shared library support (STR #17)
mxmlLoad*() now returns an error when an XML stream contains illegal control characters (STR #10)
mxmlLoad*() now returns an error when an element contains two attributes with the same name in conformance with the XML spec (STR #16)
Added support for CDATA (STR #14, STR #15)
Updated comment and processing instruction handling - no entity support per XML specification.
Added checking for invalid comment termination ("--->" is not allowed)
Changes in Mini-XML 2.1
Added support for custom data nodes (STR #6)
Now treat UTF-8 sequences which are longer than necessary as an error (STR #4)
Fixed entity number support (STR #8)
Fixed mxmlLoadString() bug with UTF-8 (STR #7)
Fixed entity lookup bug (STR #5)
Added mxmlLoadFd() and mxmlSaveFd() functions.
Fixed multi-word UTF-16 handling.
Changes in Mini-XML 2.0
New programmers manual.
Added Visual C++ project files for Microsoft Windows users.
Added optimizations to mxmldoc, mxmlSaveFile(), and mxmlIndexNew() (STR #2)
mxmlEntityAddCallback() now returns an integer status (STR #2)
Added UTF-16 support (input only; all output is UTF-8)
Added index functions to build a searchable index of XML nodes.
Added character entity callback interface to support additional character entities beyond those defined in the XHTML specification.
Added support for XHTML character entities.
The mxmldoc utility now produces XML output which conforms to an updated XML schema, described in the file "doc/mxmldoc.xsd".
Changed the whitespace callback interface to return strings instead of a single character, allowing for greater control over the formatting of XML files written using Mini-XML. THIS CHANGE WILL REQUIRE CHANGES TO YOUR 1.x CODE IF YOU USE WHITESPACE CALLBACKS.

The mxmldoc utility now produces XML output which conforms to an updated XML schema, described in the file "doc/mxmldoc.xsd".
Changed the whitespace callback interface to return strings instead of a single character, allowing for greater control over the formatting of XML files written using Mini-XML. THIS CHANGE WILL REQUIRE CHANGES TO YOUR 1.x CODE IF YOU USE WHITESPACE CALLBACKS.

The mxmldoc utility is now capable of documenting C++ classes, functions, and structures, and correctly handles C++ comments.
Added new modular tests for mxmldoc.
Updated the mxmldoc output to be more compatible with embedding in manuals produced with HTMLDOC.
The makefile incorrectly included a "/" separator between the destination path and install path. This caused problems when building and installing with MingW.
Changes in Mini-XML 1.3
Fixes for mxmldoc.
Added support for reading standard HTML entity names.
mxmlLoadString/File() did not decode character entities in element names, attribute names, or attribute values.
mxmlLoadString/File() would crash when loading non- conformant XML data under an existing parent (top) node.
Fixed several bugs in the mxmldoc utility.
Added new error callback function to catch a variety of errors and log them to someplace other than stderr.
The mxmlElementSetAttr() function now allows for NULL attribute values.
The load and save functions now properly handle quoted element and attribute name strings properly, e.g. for !DOCTYPE declarations.
Changes in Mini-XML 1.2
Added new "set" methods to set the value of a node.
Added new formatted text methods mxmlNewTextf() and mxmlSetTextf() to create/set a text node value using printf-style formats.
Added new standard callbacks for use with the mxmlLoad functions.
Updated the HTML documentation to include examples of the walk and load function output.
Added --with/without-ansi configure option to control the strdup() function check.
Added --with/without-snprintf configure option to control the snprintf() and vsnprintf() function checks.
Changes in Mini-XML 1.1.2
The mxml(3) man page wasn't updated for the string functions.
mxmlSaveString() returned the wrong number of characters.
mxml_add_char() updated the buffer pointer in the wrong place.
Changes in Mini-XML 1.1.1
The private mxml_add_ch() function did not update the start-of-buffer pointer which could cause a crash when using mxmlSaveString().
The private mxml_write_ws() function called putc() instead of using the proper callback which could cause a crash when using mxmlSaveString().
Added a mxmlSaveAllocString() convenience function for saving an XML node tree to an allocated string.
Changes in Mini-XML 1.1
The mxmlLoadFile() function now uses dynamically allocated string buffers for element names, attribute names, and attribute values. Previously they were capped at 16383, 255, and 255 bytes,

respectively.
Added a new mxmlLoadString() function for loading an XML node tree from a string.
Added a new mxmlSaveString() function for saving an XML node tree to a string.
Add emulation of strdup() if the local platform does not provide the function.
Changes in Mini-XML 1.0
The mxmldoc program now handles function arguments, structures, unions, enumerations, classes, and typedefs properly.
Documentation provided via mxmldoc and more in-line comments in the code.
Added man pages and packaging files.
Changes in Mini-XML 0.93
New mxmldoc example program that is also used to create and update code documentation using XML and produce HTML reference pages.
Added mxmlAdd() and mxmlRemove() functions to add and remove nodes from a tree. This provides more flexibility over where the nodes are inserted and allows nodes to be moved within the tree as needed.
mxmlLoadFile() now correctly handles comments.
mxmlLoadFile() now supports the required "gt", "quot", and "nbsp" character entities.
mxmlSaveFile() now uses newlines as whitespace when valid to do so.
mxmlFindElement() now also takes attribute name and attribute value string arguments to limit the search to specific elements with attributes and/or values. NULL pointers can be used as "wildcards".
Added uninstall target to makefile, and auto-reconfig if Makefile.in or configure.in are changed.
mxmlFindElement(), mxmlWalkNext(), and mxmlWalkPrev() now all provide "descend" arguments to control whether they descend into child nodes in the tree.
Fixed some whitespace issues in mxmlLoadFile().
Fixed Unicode output and whitespace issues in mxmlSaveFile().
mxmlSaveFile() now supports a whitespace callback to provide more human-readable XML output under program control.
Changes in Mini-XML 0.92
mxmlSaveFile() didn't return a value on success.
Changes in Mini-XML 0.91
mxmlWalkNext() would go into an infinite loop.
Changes in Mini-XML 0.9
Initial public release.

附录 C 库参考手册

内容
函数
mxmlAdd
mxmlDelete
mxmlElementDeleteAttr
mxmlElementGetAttr

mxmlElementSetAttr
mxmlElementSetAttrf
mxmlEntityAddCallback
mxmlEntityGetName
mxmlEntityGetValue
mxmlEntityRemoveCallback
mxmlFindElement
mxmlIndexDelete
mxmlIndexEnum
mxmlIndexFind
mxmlIndexNew
mxmlIndexReset
mxmlLoadFd
mxmlLoadFile
mxmlLoadString
mxmlNewCDATA
mxmlNewCustom
mxmlNewElement
mxmlNewInteger
mxmlNewOpaque
mxmlNewReal
mxmlNewText
mxmlNewTextf
mxmlNewXML
mxmlRelease
mxmlRemove
mxmlRetain
mxmlSAXLoadFd
mxmlSAXLoadFile
mxmlSAXLoadString
mxmlSaveAllocString
mxmlSaveFd
mxmlSaveFile
mxmlSaveString
mxmlSetCDATA
mxmlSetCustom
mxmlSetCustomHandlers
mxmlSetElement
mxmlSetErrorCallback
mxmlSetInteger
mxmlSetOpaque
mxmlSetReal
mxmlSetText
mxmlSetTextf

mxmlSetWrapMargin
mxmlWalkNext
mxmlWalkPrev
数据类型
mxml_attr_t
mxml_custom_destroy_cb_t
mxml_custom_load_cb_t
mxml_custom_save_cb_t
mxml_custom_t
mxml_element_t
mxml_error_cb_t
mxml_index_t
mxml_load_cb_t
mxml_node_t
mxml_save_cb_t
mxml_sax_cb_t
mxml_sax_event_t
mxml_text_t
mxml_value_t
结构体
mxml_attr_s
mxml_custom_s
mxml_element_s
mxml_index_s
mxml_node_s
mxml_text_s
联合
mxml_value_u
常量枚举
mxml_sax_event_e
mxml_type_e
函数
mxmlAdd
添加一个节点到树中

```
void mxmlAdd (
        mxml_node_t *parent,
        int where,
        mxml_node_t *child,
        mxml_node_t *node
);
```

参数
parent
父节点
where

添加到哪里, MXML_ADD_BEFORE or MXML_ADD_AFTER
child
where 的子节点或者使用 MXML_ADD_TO_PARENT
node
准备添加的节点
说明
添加一个指定的节点到父节点，如果 child 参数不是 NULL，将这个新的节点添加到指定的 "child" 的前面或者后面（由 where 参数决定）。如果 child 参数是 NULL，把新节点添加到子节点列表的最前面 (MXML_ADD_BEFORE) 或者时子节点列表的最后面 (MXML_ADD_AFTER)。常量 MXML_ADD_TO_PARENT 可以被用来指定一个 NULL 的 child 指针。
mxmlDelete
删除一个节点和它的所有的子节点。

```
void mxmlDelete (
    mxml_node_t *node
);
```

参数
node
被删除的节点
说明
如果这个指定的节点有一个父节点，这个函数首先使用 mxmlRemove() 函数从它的父节点中移除自己。
Mini-XML 2.4 mxmlElementDeleteAttr
删除一个参数

```
void mxmlElementDeleteAttr (
    mxml_node_t *node,
    const char *name
);
```

参数
node
XML 元素节点
name
属性名称
mxmlElementGetAttr
获取一个参数

```
const char *mxmlElementGetAttr (
    mxml_node_t *node,
    const char *name
);
```

参数
node
XML 元素节点
name
属性名称

返回值
属性值或者 NULL
说明
如果 node 参数不是一个 XML 元素或者指定的属性名不存在则返回 NULL。
mxmlElementSetAttr
设置一个属性。
```
void mxmlElementSetAttr (
    mxml_node_t *node,
    const char *name,
    const char *value
);
```
参数
node
XML 元素节点
name
属性名称
value
属性值
说明
如果这个属性名已经存在，这个属性的值将被替换为新的字符串值。这个字符串值将被拷贝到这个 XML 元素节点，如果这个节点不是一个 XML 元素，则这个函数不做任何事。
Mini-XML 2.3 mxmlElementSetAttrf
设置一个 XML 元素属性使用一个格式化的值。
```
void mxmlElementSetAttrf (
    mxml_node_t *node,
    const char *name,
    const char *format,
    ...
);
```
参数
node
XML 元素节点
name
属性名
format
"printf"风格的属性值
...
需要的附加参数(printf)风格
说明
如果这个属性名已经存在，这个属性的值将被替换为新的格式化字符串值。这个格式化后字符串值将被拷贝到这个 XML 元素节点，如果这个节点不是一个 XML 元素，则这个函数不做任何事。
mxmlEntityAddCallback
添加一个回调函数来将 XML 实体转换为 Unicode 编码字符。

int mxmlEntityAddCallback (void);
返回值
0 成功, -1 失败
回调函数用来将类似于"&"之类的 XML 实体字符串转化为用户设定的 Unicode 编码。此函数的原型应该为：int mxmlEntityAddCallback(int (*cb)(const char *name)) 参数为自定义中的回调函数。目前仅支持将实体转换为单个字符的转换，而不支持实体字符串。 Z.F
mxmlEntityGetName
获取一个字符值对应的 XML 实体名字。
const char *mxmlEntityGetName (
    int val
);
参数
val
字符值
返回值
XML 实体名字或者 NULL
说明
如果 val 不需要被标识为一个命名的 XML 实体，返回 NULL。
如：val = '&'，将返回"&" Z.F
mxmlEntityGetValue
获取一个代表到一个 XML 命名实体的字符。
int mxmlEntityGetValue (
    const char *name
);
参数
name
XML 实体名字
返回值
字符值或者-1 代表错误
说明
XML 实体名字总是可以被关联到一个数字常量，如果这个名字未知则返回-1。
mxmlEntityRemoveCallback
删除一个 XML 实体回调。
void mxmlEntityRemoveCallback (void);
函数原型应该是：void mxmlEntityRemoveCallback(int (*cb)(const char *name)); Z.F
mxmlFindElement
搜索一个命名的 XML 元素。
mxml_node_t *mxmlFindElement (
    mxml_node_t *node,
    mxml_node_t *top,
    const char *name,
    const char *attr,
    const char *value,
    int descend

);
参数
node
当前节点
top
顶级节点
name
XML 元素名，或者 NULL 匹配所有元素
attr
属性名，或者 NULL 表示不匹配属性
value
属性值，或者 NULL 表示任何值
descend
在 XML 树中向下搜索模式： MXML_DESCEND, MXML_NO_DESCEND, 或者 MXML_DESCEND_FIRST
返回值
XMl 元素节点或者 NULL
说明
搜索可以被 XML 元素名，属性名和属性值所限定；任何名字或者值等于 NULL 被处理就相当于通配符，所以使用不同的搜索方法可以被实现用来查看所有的指定名称的 XML 元素或者是所有的具有指定属性的 XML 元素。参数： descend 确定了是否向下搜索子节点；通常你将使用 MXML_DESCEND_FIRST 作为第一次搜索，然后使用使用 MXML_NO_DESCEND 来发现更多的这个节点的直接子节点。 top 节点参数约束了搜索在一个指定节点的子节点中。
mxmlIndexDelete
删除一个索引。
void mxmlIndexDelete (
    mxml_index_t *ind
);
参数
ind
被删除的索引
mxmlIndexEnum
返回索引中的下一个节点。
mxml_node_t *mxmlIndexEnum (
    mxml_index_t *ind
);
参数
ind
进行枚举的索引
返回值
下一个节点或者 NULL 代表没有更多的节点
说明
返回节点顺序将按照索引的排序被返回。

mxmlIndexFind
搜索下一个匹配的节点。
```
mxml_node_t *mxmlIndexFind (
    mxml_index_t *ind,
    const char *element,
    const char *value
);
```
参数
ind
进行搜索的索引
element
如不为 NULL，代表想要搜索的 XML 元素名。
value
如不为 NULL，代表想要搜索的属性值。
返回值
节点或者 NULL 代表没有发现。
说明
你在第一次使用一个特定的包含"element"和"value"字符串的集合来调用这个函数之前应该首先调用 mxmlIndexReset()函数。如果"element"和"value"同时等于 NULL 则相当于调用了 mxmlIndexEnum()函数。
mxmlIndexNew
创建一个新的索引。
```
mxml_index_t *mxmlIndexNew (
    mxml_node_t *node,
    const char *element,
    const char *attr
);
```
参数
node
XML 节点树
element
索引的 XML 元素名或者 NULL 代表所有元素
attr
索引的 XML 属性名或者 NULL 代表不使用。
返回值
新的索引
说明
被创建的索引将包含具备指定的元素名和/或属性所有的节点。如果"element" 和"attr"同时等于 NULL，索引将包含一个被排序的完整节点树的列表。节点被按照 XML 元素名和选择的属性值（如果"attr"参数不等于 NULL）进行排序。
mxmlIndexReset
重设索引中的枚举/搜索指针并且返回索引中的第一个节点。
```
mxml_node_t *mxmlIndexReset (
    mxml_index_t *ind
```

);
参数
ind
准备重设的索引
返回值
第一个节点或者 NULL 代表索引为空。
说明
这个函数需要被首先调用，在第一次使用函数 mxmlIndexEnum() 或 mxmlIndexFind()之前。
mxmlLoadFd
载入一个文件描述符到一个 XML 节点树。
```
mxml_node_t *mxmlLoadFd (
    mxml_node_t *top,
    int fd,
    mxml_load_cb_t cb
);
```
参数
top
顶部节点
fd
需要进行读取的文件描述符
cb
回调函数或者 MXML_NO_CALLBACK
返回值
第一个节点或者 NULL 代表文件不能被读取。
说明
在指定文件中的所有节点将被添加到所指定的顶部节点。如果没有"top"顶部节点被提供，这个 XML 文件必须是规范的并且整个文件只有一个父节点为<?xml> 。 回调函数返回的值类型将被使用到子节点。如果 MXML_NO_CALLBACK 参数被指定，那么所有的子节点将都会是 MXML_ELEMENT 或者 MXML_TEXT 其中之一的节点。
常 量 MXML_INTEGER_CALLBACK, MXML_OPAQUE_CALLBACK, MXML_REAL_CALLBACK, 和 MXML_TEXT_CALLBACK 定义了将载入指定类型的子节点。
mxmlLoadFile
载入一个文件到一个 XML 节点树。
```
mxml_node_t *mxmlLoadFile (
    mxml_node_t *top,
    FILE *fp,
    mxml_load_cb_t cb
);
```
参数
top
顶级节点
fp
准备读取的文件

cb
回调函数或 MXML_NO_CALLBACK
返回值
第一个节点或者 NULL 代表文件不能被读取。
说明
在指定文件中的所有节点将被添加到所指定的顶部节点。如果没有"top"顶部节点被提供，这个 XML 文件必须是规范的并且整个文件只有一个父节点为<?xml> 。 回调函数返回的值类型将被使用到子节点。如果 MXML_NO_CALLBACK 参数被指定，那么所有的子节点将都会是 MXML_ELEMENT 或者 MXML_TEXT 其中之一的节点.
常 量 MXML_INTEGER_CALLBACK, MXML_OPAQUE_CALLBACK, MXML_REAL_CALLBACK, 和 MXML_TEXT_CALLBACK 定义了将载入指定类型的子节点。
mxmlLoadString
载入一个文件到一个 XML 节点树。
```
mxml_node_t *mxmlLoadString (
    mxml_node_t *top,
    const char *s,
    mxml_load_cb_t cb
);
```
参数
top
顶级节点
s
准备读取的字符串
cb
回调函数或 MXML_NO_CALLBACK
返回值
第一个节点或者 NULL 代表字符串中有错误。
说明
在指定字符串中的所有节点将被添加到所指定的顶部节点。如果没有"top"顶部节点被提供，这个 XML 字符串必须是规范的并且整个文件只有一个父节点为<?xml> 。 回调函数返回的值类型将被使用到子节点。如果 MXML_NO_CALLBACK 参数被指定，那么所有的子节点将都会是 MXML_ELEMENT 或者 MXML_TEXT 其中之一的节点.
T 常 量 MXML_INTEGER_CALLBACK, MXML_OPAQUE_CALLBACK, MXML_REAL_CALLBACK, 和 MXML_TEXT_CALLBACK 定义了将载入指定类型的子节点。
Mini-XML 2.3 mxmlNewCDATA
创建一个新的 CDATA 节点.
```
mxml_node_t *mxmlNewCDATA (
    mxml_node_t *parent,
    const char *data
);
```
参数
parent

父节点或者 MXML_NO_PARENT
data
数据字符串
返回值
新的节点
说明
新的 CDATA 节点将被添加到指定父节点的子节点列表的最后，常量 MXML_NO_PARENT 可以被用来指定新的 CDATA 节点没有父节点。数据字符串必须是以空字符结尾，并被拷贝到新的 CDATA 节点。CDATA 节点使用 MXML_ELEMENT 节点类型。
Mini-XML 2.1 mxmlNewCustom
创建一个新的用户自定义数据节点。

```
mxml_node_t *mxmlNewCustom (
    mxml_node_t *parent,
    void *data,
    mxml_custom_destroy_cb_t destroy
);
```

参数
parent
父节点或者 MXML_NO_PARENT
data
指向数据的指针
destroy
销毁数据使用的函数
返回值
新节点
说明
新的自定义节点将被添加到指定父节点的子节点列表的最后。常量 MXML_NO_PARENT 可以被用来指定新的自定义节点没有父节点。NULL 可以被通过，当数据节点不是动态分配或者是独立管理时。
最后一句 NULL 应该是指参数：destroy，表示不使用销毁函数。 Z.F
mxmlNewElement
创建一个新的 XML 元素节点。

```
mxml_node_t *mxmlNewElement (
    mxml_node_t *parent,
    const char *name
);
```

参数
parent
父节点或 MXML_NO_PARENT
name
XML 元素名称
返回值
新节点
说明

新的 XML 元素节点将被添加到指定父节点的子节点列表的最后。常量 MXML_NO_PARENT 可以被用来指定新的 XML 元素节点没有父节点。

mxmlNewInteger

创建一个新的整数节点。

```
mxml_node_t *mxmlNewInteger (
    mxml_node_t *parent,
    int integer
);
```

参数

parent

父节点或 MXML_NO_PARENT

integer

整形值

返回值

新节点

说明

新的整数节点将被添加到指定父节点的子节点列表的最后。常量 MXML_NO_PARENT 可以被用来指定新的整数节点没有父节点。

mxmlNewOpaque

创建一个新的不透明字符串节点

```
mxml_node_t *mxmlNewOpaque (
    mxml_node_t *parent,
    const char *opaque
);
```

参数

parent

父节点或 MXML_NO_PARENT

opaque

不透明字符串

返回值

新节点

说明

新的不透明字符串节点将被添加到指定父节点的子节点列表的最后。常量 MXML_NO_PARENT 可以被用来指定新的不透明字符串节点没有父节点。这个字符串必须是空字符结尾并被拷贝到新节点。

mxmlNewReal

创建一个新的浮点数节点。

```
mxml_node_t *mxmlNewReal (
    mxml_node_t *parent,
    double real
);
```

参数

parent

父节点或 MXML_NO_PARENT

real
浮点数值
返回值
新节点
说明
新的浮点数节点将被添加到指定父节点的子节点列表的最后。常量 MXML_NO_PARENT 可以被用来指定新的浮点数节点没有父节点。
mxmlNewText
创建新的文本分段节点。

```
mxml_node_t *mxmlNewText (
    mxml_node_t *parent,
    int whitespace,
    const char *string
);
```

参数
parent
父节点或 MXML_NO_PARENT
whitespace
1 = 有前导空格, 0 = 没有空格
string
字符串
返回值
新节点
说明
新的文本节点将被添加到指定父节点的子节点列表的最后。常量 MXML_NO_PARENT 可以被用来指定新的文本节点没有父节点。参数：whitespace 被用在指定是否在这个节点前面有前导空格。文本字符串必须时以空字符结尾并被拷贝到新的节点。
mxmlNewTextf
创建一个新的格式化的文本分段节点

```
mxml_node_t *mxmlNewTextf (
    mxml_node_t *parent,
    int whitespace,
    const char *format,
    ...
);
```

参数
parent
父节点或 MXML_NO_PARENT
whitespace
1 = 有前导空格, 0 = 没有空格
format
"printf"风格的格式化字符串
...
需要的附加参数

返回值
新节点
说明
新的文本节点将被添加到指定父节点的子节点列表的最后。常量 MXML_NO_PARENT 可以被用来指定新的文本节点没有父节点。参数：whitespace 被用在指定是否在这个节点前面有前导空格。格式化字符串必须时以空字符结尾并被格式化到新的节点。

Mini-XML 2.3 mxmlNewXML

创建一个新的 XML 文档树。

```
mxml_node_t *mxmlNewXML (
    const char *version
);
```

参数
version
使用的版本号
返回值
新的 "?xml" 节点
说明
参数 "version" 指定了放在"?xml" 元素节点中的版本号。如果为 NULL 则假定为 "version 1.0"。

Mini-XML 2.3 mxmlRelease

释放一个节点。

```
int mxmlRelease (
    mxml_node_t *node
);
```

参数
node
节点
返回值
新的引用计数
说明
当引用计数为 0 时，这个节点（以及所有子节点）被通过函数 mxmlDelete() 所删除。

mxmlRemove

移除一个节点从它的父节点中。

```
void mxmlRemove (
    mxml_node_t *node
);
```

参数
node
被移除的节点
说明
不释放节点使用的内存，使用函数 mxmlDelete() 来释放。如果这个节点没有父节点则这个函数不做任何事。

Mini-XML 2.3 mxmlRetain

保留一个节点

```
int mxmlRetain (
    mxml_node_t *node
);
```
参数
node
节点
返回值
新的引用计数
Mini-XML 2.3 mxmlSAXLoadFd
使用 SAX 回调从一个文件描述符中加载数据到一个 XML 节点树。
```
mxml_node_t *mxmlSAXLoadFd (
    mxml_node_t *top,
    int fd,
    mxml_load_cb_t cb,
    mxml_sax_cb_t sax_cb,
    void *sax_data
);
```
参数
top
顶级节点
fd
进行读取的文件描述符
cb
XML 节点类型回调函数或者 MXML_NO_CALLBACK
sax_cb
SAX 回调函数或者 MXML_NO_CALLBACK
sax_data
SAX 用户数据
返回值
第一个节点或者 NULL 代表文件不能被读取。
说明
在指定文件中的节点将被添加到指定的顶级节点中。如果"top"节点没有提供，这个 XML 文件必须是规范的并且整个文件只有一个父节点为<?xml> 。回调函数"cb"返回子节点的值类型。如果 MXML_NO_CALLBACK 参数被指定，那么所有的子节点将都会是 MXML_ELEMENT 或者 MXML_TEXT 其中之一的节点。
常量 MXML_INTEGER_CALLBACK, MXML_OPAQUE_CALLBACK, MXML_REAL_CALLBACK, 和 MXML_TEXT_CALLBACK 被定义用于加载指定类型的子节点。
在 SAX 回调函数中("sax_cb")，对于所有节点都必须调用 mxmlRetain()函数用于保留为以后使用。否则，节点将在父节点被关闭时或者到达数据、注释、CDATA 和指令节点时被删除。
Mini-XML 2.3 mxmlSAXLoadFile
使用 SAX 回调从一个文件中加载数据到一个 XML 节点树。
```
mxml_node_t *mxmlSAXLoadFile (
    mxml_node_t *top,
```

```
    FILE *fp,
    mxml_load_cb_t cb,
    mxml_sax_cb_t sax_cb,
    void *sax_data
);
```

参数

top

顶级节点

fp

进行读取的文件

cb

XML 节点类型回调函数或者 MXML_NO_CALLBACK

sax_cb

SAX 回调函数或者 MXML_NO_CALLBACK

sax_data

SAX 用户数据

返回值

第一个节点或者 NULL 代表文件不能被读取。

说明

在指定文件中的节点将被添加到指定的顶级节点中。如果"top"节点没有提供，这个 XML 文件必须是规范的并且整个文件只有一个父节点为<?xml> 。回调函数"cb"返回子节点的值类型。如果 MXML_NO_CALLBACK 参数被指定，那么所有的子节点将都会是 MXML_ELEMENT 或者 MXML_TEXT 其中之一的节点。

常量 MXML_INTEGER_CALLBACK, MXML_OPAQUE_CALLBACK, MXML_REAL_CALLBACK, 和 MXML_TEXT_CALLBACK 被定义用于加载指定类型的子节点。

在 SAX 回调函数中("sax_cb")，对于所有节点都必须调用 mxmlRetain()函数用于保留为以后使用。否则，节点将在父节点被关闭时或者到达数据、注释、CDATA 和指令节点时被删除。

Mini-XML 2.3 mxmlSAXLoadString

使用 SAX 回调从一个字符串中加载数据到一个 XML 节点树。

```
mxml_node_t *mxmlSAXLoadString (
    mxml_node_t *top,
    const char *s,
    mxml_load_cb_t cb,
    mxml_sax_cb_t sax_cb,
    void *sax_data
);
```

参数

top

顶级节点

s

准备加载的字符串

cb

XML 节点类型回调函数或者 MXML_NO_CALLBACK

sax_cb
SAX 回调函数或者 MXML_NO_CALLBACK
sax_data
SAX 用户数据
返回值
第一个节点或者 NULL 代表文件不能被读取。
说明
在指定字符串中的节点将被添加到指定的顶级节点中。如果"top"节点没有提供，这个 XML 字符串必须是规范的并且整个文件只有一个父节点为<?xml> 。回调函数"cb"返回子节点的值类型。如果 MXML_NO_CALLBACK 参数被指定，那么所有的子节点将都会是 MXML_ELEMENT 或者 MXML_TEXT 其中之一的节点。
常量 MXML_INTEGER_CALLBACK, MXML_OPAQUE_CALLBACK, MXML_REAL_CALLBACK, 和 MXML_TEXT_CALLBACK 被定义用于加载指定类型的子节点。
在 SAX 回调函数中("sax_cb")，对于所有节点都必须调用 mxmlRetain()函数用于保留为以后使用。否则，节点将在父节点被关闭时或者到达数据、注释、CDATA 和指令节点时被删除。
mxmlSaveAllocString
保存一个 XML 节点树到一个内部分配的字符串。

```
char *mxmlSaveAllocString (
    mxml_node_t *node,
    mxml_save_cb_t cb
);
```

参数
node
准备要写入的节点
cb
空白回调函数或者 MXML_NO_CALLBACK
返回值
分配的字符串或者 NULL
说明
这个函数返回一个指向字符串的指针包含了描述整个 XML 节点树的文本。当你使用完这个字符串后需要使用 free()函数来释放。如果这个节点产生了一个空字符或者字符串分配失败将返回 NULL。
回调函数参数指定了一个函数用来在每个 XML 元素之前或者之后返回一个空白字符串或者 NULL。如果指定了 MXML_NO_CALLBACK，空格将仅被添加到具有前导空格的 MXML_TEXT 节点前面（node->value->text->whitespace = 1）和在一个打开 XML 元素标签的属性名称前面。
mxmlSaveFd
保存一个 XML 节点树到一个文件描述符。

```
int mxmlSaveFd (
    mxml_node_t *node,
    int fd,
    mxml_save_cb_t cb
);
```

参数
node
准备要写入的节点
fd
准备写入的文件描述符
cb
空白回调函数或者 MXML_NO_CALLBACK
返回值
成功返回 0，错误返回 -1。
说明
回调函数参数指定了一个函数用来在每个 XML 元素之前或者之后返回一个空白字符串或者 NULL。如果指定了 MXML_NO_CALLBACK，空格将仅被添加到具有前导空格的 MXML_TEXT 节点前面（node->value->text->whitespace = 1）和在一个打开 XML 元素标签的属性名称前面。
mxmlSaveFile
保存一个 XML 节点树到一个文件。

```
int mxmlSaveFile (
    mxml_node_t *node,
    FILE *fp,
    mxml_save_cb_t cb
);
```

参数
node
准备要写入的节点
fp
准备写入的文件。
cb
空白回调函数或者 MXML_NO_CALLBACK
返回值
成功返回 0，错误返回 -1。
说明
回调函数参数指定了一个函数用来在每个 XML 元素之前或者之后返回一个空白字符串或者 NULL。如果指定了 MXML_NO_CALLBACK，空格将仅被添加到具有前导空格的 MXML_TEXT 节点前面（node->value->text->whitespace = 1）和在一个打开 XML 元素标签的属性名称前面。
mxmlSaveString
保存一个 XML 节点树到一个字符串。

```
int mxmlSaveString (
    mxml_node_t *node,
    char *buffer,
    int bufsize,
    mxml_save_cb_t cb
);
```

参数

node
准备要写入的节点
buffer
字符串缓冲区
bufsize
字符串缓冲区大小
cb
空白回调函数或者 MXML_NO_CALLBACK
返回值
字符串大小
说明
这个函数返回字符串需要字节总数，但是最多拷贝(bufsize-1)个字符到指定的 buffer 中。
如：一个 XML 树字符串长度为 200 个字节，但缓冲区只有 100 字节，则返回 200，但只拷贝了 99 个字符，最后一个为空字符。Z.F
回调函数参数指定了一个函数用来在每个 XML 元素之前或者之后返回一个空白字符串或者 NULL。如果指定了 MXML_NO_CALLBACK，空格将仅被添加到具有前导空格的 MXML_TEXT 节点前面（node->value->text->whitespace = 1）和在一个打开 XML 元素标签的属性名称前面。

Mini-XML 2.3 mxmlSetCDATA

设置一个 CDATA 元素节点的名称。

```
int mxmlSetCDATA (
    mxml_node_t *node,
    const char *data
);
```

参数
node
准备设置的节点
data
新的数据字符串
返回值
成功返回 0，失败返回 -1。
说明
如果这个节点不是一个 CDATA 节点则节点不发生改变。

Mini-XML 2.1 mxmlSetCustom

对一个用户自定义数据节点设置数据和销毁回调函数。

```
int mxmlSetCustom (
    mxml_node_t *node,
    void *data,
    mxml_custom_destroy_cb_t destroy
);
```

参数
node
被设置的节点
data

新的数据指针
destroy
新的销毁回调函数
返回值
成功返回 0，失败返回 -1。
说明
如果这个节点不是一个用户自定义节点则节点不发生改变。
mxmlSetCustomHandlers
设置对于自定义数据的处理回调函数。
```
void mxmlSetCustomHandlers (
    mxml_custom_load_cb_t load,
    mxml_custom_save_cb_t save
);
```
参数
load
加载回调函数
save
保存回调函数
说明
加载回调函数接收一个节点指针和数据字符串，成功时必须返回 0，错误时返回一个非 0 值。
保存回调函数接收一个节点指针，成功时必须返回一个使用 malloc 分配的字符串，错误时返回 NULL。
mxmlSetElement
设置 XML 元素节点的名字。
```
int mxmlSetElement (
    mxml_node_t *node,
    const char *name
);
```
参数
node
被设置的节点
name
新的名称字符串
返回值
成功返回 0，失败返回 -1。
说明
如果这个节点不是一个 XML 元素节点则节点不发生改变。
mxmlSetErrorCallback
设置错误信息回调函数。
```
void mxmlSetErrorCallback (
    mxml_error_cb_t cb
);
```
参数

cb
错误回调函数
mxmlSetInteger
设置一个整数节点的值。
```
int mxmlSetInteger (
        mxml_node_t *node,
        int integer
);
```
参数
node
被设置的节点
integer
整数值
返回值
成功返回 0，失败返回 -1。
说明
如果这个节点不是一个整数节点则节点不发生改变。
mxmlSetOpaque
设置一个不透明字符串节点的值。
```
int mxmlSetOpaque (
        mxml_node_t *node,
        const char *opaque
);
```
参数
node
被设置的节点
opaque
不透明字符串
返回值
成功返回 0，失败返回 -1。
说明
如果这个节点不是一个不透明字符串节点则节点不发生改变。
mxmlSetReal
设置一个浮点数节点的值。
```
int mxmlSetReal (
        mxml_node_t *node,
        double real
);
```
参数
node
被设置的节点
real
浮点数值
返回值

成功返回 0，失败返回 -1。
说明
如果这个节点不是一个浮点数节点则节点不发生改变。
mxmlSetText
设置一个文本节点的值。
```
int mxmlSetText (
        mxml_node_t *node,
        int whitespace,
        const char *string
);
```
参数
node
被设置的节点
whitespace
1 = 有前导空格, 0 = 没有前导空格
string
字符串
返回值
成功返回 0，失败返回 -1。
说明
如果这个节点不是一个文本节点则节点不发生改变。
mxmlSetTextf
设置一个文本节点的值为一个格式化的字符串。
```
int mxmlSetTextf (
        mxml_node_t *node,
        int whitespace,
        const char *format,
        ...
);
```
参数
node
被设置的节点
whitespace
1 = 有前导空格, 0 = 没有前导空格
format
"printf"风格的格式化字符串
...
需要的附加参数
返回值
成功返回 0，失败返回 -1。
说明
如果这个节点不是一个文本节点则节点不发生改变。
Mini-XML 2.3 mxmlSetWrapMargin
设置在保存 XML 数据时的自动折行位置。

```
void mxmlSetWrapMargin (
    int column
);
```
参数
column
自动折行的列
说明
当"column" is <= 0 时取消自动折行。
mxmlWalkNext
遍历到 XML 树中的下一个逻辑节点。
```
mxml_node_t *mxmlWalkNext (
    mxml_node_t *node,
    mxml_node_t *top,
    int descend
);
```
参数
node
当前节点
top
顶级节点
descend
在 XML 树中的向下搜索模式 - MXML_DESCEND, MXML_NO_DESCEND, 或者 MXML_DESCEND_FIRST。
返回值
下一个节点或者 NULL
说明
"descend"参数控制下一个节点是否考虑第一个子节点。"top"参数约束了遍历这个节点的所有子节点。
mxmlWalkPrev
遍历到 XML 树中的上一个逻辑节点。
```
mxml_node_t *mxmlWalkPrev (
    mxml_node_t *node,
    mxml_node_t *top,
    int descend
);
```
参数
node
当前节点
top
顶级节点
descend
在 XML 树中的向下搜索模式 - MXML_DESCEND, MXML_NO_DESCEND, 或者 MXML_DESCEND_FIRST。
返回值

上一个节点或者 NULL
说明
"descend"参数控制下一个节点是否考虑第一个子节点。"top"参数约束了遍历这个节点的所有子节点。
数据类型
mxml_attr_t
XML 元素节点的属性值。
typedef struct mxml_attr_s mxml_attr_t;
mxml_custom_destroy_cb_t
自定义数据销毁回调函数原型
typedef void (*mxml_custom_destroy_cb_t)(void *);
mxml_custom_load_cb_t
自定义数据加载回调函数原型
typedef int (*mxml_custom_load_cb_t)(mxml_node_t *, const char *);
mxml_custom_save_cb_t
自定义数据保存回调函数原型
typedef char *(*mxml_custom_save_cb_t)(mxml_node_t *);
Mini-XML 2.1 mxml_custom_t
自定义 XML 类型值
typedef struct mxml_custom_s mxml_custom_t;
mxml_element_t
XML 元素值
typedef struct mxml_element_s mxml_element_t;
mxml_error_cb_t
错误回调函数原型
typedef void (*mxml_error_cb_t)(const char *);
mxml_index_t
XML 节点索引
typedef struct mxml_index_s mxml_index_t;
mxml_load_cb_t
加载回调函数
typedef mxml_type_t (*mxml_load_cb_t)(mxml_node_t *);
mxml_node_t
XML 节点
typedef struct mxml_node_s mxml_node_t;
mxml_save_cb_t
保存回调函数
typedef const char *(*mxml_save_cb_t)(mxml_node_t *, int);
mxml_sax_cb_t
SAX 回调函数
typedef void (*mxml_sax_cb_t)(mxml_node_t *, mxml_sax_event_t, void *);
mxml_sax_event_t
SAX 事件类型.
typedef enum mxml_sax_event_e mxml_sax_event_t;

mxml_text_t
XML 文本节点值
typedef struct mxml_text_s mxml_text_t;
mxml_value_t
XML 节点值
typedef union mxml_value_u mxml_value_t;
Structures
mxml_attr_s
XML 元素的属性值
```
struct mxml_attr_s {
        char *name;
        char *value;
};
```
成员
name
属性名
value
属性值
Mini-XML 2.1 mxml_custom_s
自定义 XML 节点值
```
struct mxml_custom_s {
        void *data;
        mxml_custom_destroy_cb_t destroy;
};
```
成员
data
指向一个自定义数据的指针（已分配的）。
destroy
指向销毁回调函数的指针
mxml_element_s
XML 元素值
```
struct mxml_element_s {
        mxml_attr_t *attrs;
        char *name;
        int num_attrs;
};
```
成员
attrs
包含的所有属性
name
XML 元素名称
num_attrs
包含的属性数量
mxml_index_s

XML 节点索引

```
struct mxml_index_s {
        int alloc_nodes;
        char *attr;
        int cur_node;
        mxml_node_t **nodes;
        int num_nodes;
};
```

成员

alloc_nodes

在索引中的已分配的节点数

attr

节点使用的属性或者 NULL

cur_node

当前节点

nodes

包含的节点数组

num_nodes

在索引中的节点总数

mxml_node_s

XML 节点。

```
struct mxml_node_s {
        struct mxml_node_s *child;
        struct mxml_node_s *last_child;
        struct mxml_node_s *next;
        struct mxml_node_s *parent;
        struct mxml_node_s *prev;
        int ref_count;
        mxml_type_t type;
        void *user_data;
        mxml_value_t value;
};
```

成员

child

第一个子节点

last_child

最后一个子节点

next

同级的下一个节点（在同一个父节点下）

parent

父节点

prev

同级的上一个节点（在同一个父节点下）

ref_count

使用计数器（引用计数）
type
节点类型
user_data
用户关联数据
value
节点值
mxml_text_s
XML 文本节点值
```
struct mxml_text_s {
        char *string;
        int whitespace;
};
```
成员
string
字符串片断
whitespace
是否有前导空格?
联合
mxml_value_u
XML 节点值。
```
union mxml_value_u {
        mxml_custom_t custom;
        mxml_element_t element;
        int integer;
        char *opaque;
        double real;
        mxml_text_t text;
};
```
成员
custom   Mini-XML 2.1
自定义数据
element
元素
integer
整数
opaque
不透明字符串
real
浮点数
text
文本片断
常量
mxml_sax_event_e

SAX 事件类型。
常量
MXML_SAX_CDATA
CDATA 节点
MXML_SAX_COMMENT
注释节点
MXML_SAX_DATA
数据节点
MXML_SAX_DIRECTIVE
处理指令节点
MXML_SAX_ELEMENT_CLOSE
XML 元素关闭节点
MXML_SAX_ELEMENT_OPEN
XML 元素开放节点
mxml_type_e
XML 节点类型
常量
MXML_CUSTOM Mini-XML 2.1
自定义数据
MXML_ELEMENT
XML 元素并包含属性
MXML_IGNORE Mini-XML 2.3
忽略/抛弃的节点
MXML_INTEGER
整数值
MXML_OPAQUE
透明字符串值
MXML_REAL
浮点数值
MXML_TEXT
文本片断

附录 D XML Schema

这个附录提供了 mxmldoc 程序生成 XML 文件时使用的 XML schema。这个 schema 的在线版本参见：

http://www.easysw.com/~mike/mxmldoc.xsd

mxmldoc.xsd

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      Mini-XML 2.3 documentation schema for mxmldoc output.
```

```
        Copyright 2003-2007 by Michael Sweet.
    </xsd:documentation>
  </xsd:annotation>
  <!-- basic element definitions -->
  <xsd:element name="argument" type="argumentType"/>
  <xsd:element name="class" type="classType"/>
  <xsd:element name="constant" type="constantType"/>
  <xsd:element name="description" type="xsd:string"/>
  <xsd:element name="enumeration" type="enumerationType"/>
  <xsd:element name="function" type="functionType"/>
  <xsd:element name="mxmldoc" type="mxmldocType"/>
  <xsd:element name="namespace" type="namespaceType"/>
  <xsd:element name="returnvalue" type="returnvalueType"/>
  <xsd:element name="seealso" type="identifierList"/>
  <xsd:element name="struct" type="structType"/>
  <xsd:element name="typedef" type="typedefType"/>
  <xsd:element name="type" type="xsd:string"/>
  <xsd:element name="union" type="unionType"/>
  <xsd:element name="variable" type="variableType"/>
  <!-- descriptions of complex elements -->
  <xsd:complexType name="argumentType">
    <xsd:sequence>
      <xsd:element ref="type" minOccurs="1" maxOccurs="1"/>
      <xsd:element ref="description" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="default" type="xsd:string" use="optional"/>
    <xsd:attribute name="name" type="identifier" use="required"/>
    <xsd:attribute name="direction" type="direction" use="optional"
     default="I"/>
  </xsd:complexType>
  <xsd:complexType name="classType">
    <xsd:sequence>
      <xsd:element ref="description" minOccurs="0" maxOccurs="1"/>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
<xsd:element ref="class"/>
<xsd:element ref="enumeration"/>
<xsd:element ref="function"/>
<xsd:element ref="struct"/>
<xsd:element ref="typedef"/>
<xsd:element ref="union"/>
<xsd:element ref="variable"/>
      </xsd:choice>
    </xsd:sequence>
    <xsd:attribute name="name" type="identifier" use="required"/>
```

```
    <xsd:attribute name="parent" type="xsd:string" use="optional"/>
  </xsd:complexType>
  <xsd:complexType name="constantType">
    <xsd:sequence>
      <xsd:element ref="description" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="identifier" use="required"/>
  </xsd:complexType>
  <xsd:complexType name="enumerationType">
    <xsd:sequence>
      <xsd:element ref="description" minOccurs="0" maxOccurs="1"/>
      <xsd:element ref="constant" minOccurs="1" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="identifier" use="required"/>
  </xsd:complexType>
  <xsd:complexType name="functionType">
    <xsd:sequence>
      <xsd:element ref="returnvalue" minOccurs="0" maxOccurs="1"/>
      <xsd:element ref="description" minOccurs="0" maxOccurs="1"/>
      <xsd:element ref="argument" minOccurs="1" maxOccurs="unbounded"/>
      <xsd:element ref="seealso" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="identifier" use="required"/>
    <xsd:attribute name="scope" type="scope" use="optional"/>
  </xsd:complexType>
  <xsd:complexType name="mxmldocType">
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
      <xsd:element ref="class"/>
      <xsd:element ref="enumeration"/>
      <xsd:element ref="function"/>
      <xsd:element ref="namespace"/>
      <xsd:element ref="struct"/>
      <xsd:element ref="typedef"/>
      <xsd:element ref="union"/>
      <xsd:element ref="variable"/>
    </xsd:choice>
  </xsd:complexType>
  <xsd:complexType name="namespaceType">
    <xsd:sequence>
      <xsd:element ref="description" minOccurs="0" maxOccurs="1"/>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
<xsd:element ref="class"/>
<xsd:element ref="enumeration"/>
<xsd:element ref="function"/>
```

```
<xsd:element ref="struct"/>
<xsd:element ref="typedef"/>
<xsd:element ref="union"/>
<xsd:element ref="variable"/>
        </xsd:choice>
    </xsd:sequence>
    <xsd:attribute name="name" type="identifier" use="required"/>
 </xsd:complexType>
 <xsd:complexType name="returnvalueType">
    <xsd:sequence>
        <xsd:element ref="type" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="description" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
 </xsd:complexType>
 <xsd:complexType name="structType">
    <xsd:sequence>
        <xsd:element ref="description" minOccurs="0" maxOccurs="1"/>
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
<xsd:element ref="variable"/>
<xsd:element ref="function"/>
        </xsd:choice>
    </xsd:sequence>
    <xsd:attribute name="name" type="identifier" use="required"/>
 </xsd:complexType>
 <xsd:complexType name="typedefType">
    <xsd:sequence>
        <xsd:element ref="type" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="description" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="identifier" use="required"/>
 </xsd:complexType>
 <xsd:complexType name="unionType">
    <xsd:sequence>
        <xsd:element ref="description" minOccurs="0" maxOccurs="1"/>
        <xsd:element ref="variable" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="identifier" use="required"/>
 </xsd:complexType>
 <xsd:complexType name="variableType">
    <xsd:sequence>
        <xsd:element ref="type" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="description" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="identifier" use="required"/>
```

```
    </xsd:complexType>
    <!-- data types -->
    <xsd:simpleType name="direction">
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="I"/>
        <xsd:enumeration value="O"/>
        <xsd:enumeration value="IO"/>
      </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="identifier">
      <xsd:restriction base="xsd:string">
        <xsd:pattern value="[a-zA-Z_(.]([a-zA-Z_(.,)* 0-9])*"/>
      </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="identifierList">
      <xsd:list itemType="identifier"/>
    </xsd:simpleType>
    <xsd:simpleType name="scope">
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value=""/>
        <xsd:enumeration value="private"/>
        <xsd:enumeration value="protected"/>
        <xsd:enumeration value="public"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:schema>
```