<https://www.cnblogs.com/zhanghaifeng/archive/2012/06/05/2536613.html#top>

摘要

Libxml是个有免费许可的用于处理XML、能轻松跨越多个平台的C语言库。这个指南提供他的基本函数的例子。

绪论

Libxml是个实现读、创建及操纵XML数据功能的C语言库。这个指南提供例子代码并给出他基本功能的解释。在这个项目的主页上有Libxml及更多关于他可用的资料。包含有完整的API文件。这个指南并不能替代这些完整的文件，不过阐明功能需要使用库来完成基本操作。

   这个指南基于一个简单的XML应用，他使用我写的一篇文章生成，他包含有元数据和文章的主体。

本指南中的例子代码示范怎么做到：

&#8226; 解析文件

&#8226; 取得指定元素的文本

&#8226; 添加一个元素及他的内容

&#8226; 添加一个属性

&#8226; 取得一个属性的值

例子的完整代码包含在附录中

数据类型

Libxml定义了许多数据类型，我们将反复碰到他们，他隐藏了杂乱的来源以致你不必处理他除非你有特定的需要。xmlChar  替代char,使用UTF-8编码的一字节字符串。如果你的数据使用其他编码，他必须被转换到UTF-8才能使用libxml的函数。在libxml编码支持WEB页面有更多关于编码的有用信息。

XmlDoc 包含由解析文件建立的树结构，xmlDocPtr是指向这个结构的指针。

xmlNodePtr and xmlNode 包含单一结点的结构xmlNodePtr是指向这个结构的指针，他被用于遍历文件树。

解析文件

解析文件时仅仅需要文件名并只调用一个函数，并有错误检查。完整代码：附录C, Keyword例程代码

①xmlDocPtr doc;

②xmlNodePtr cur;

③doc = xmlParseFile(docname);

④if (doc == NULL ) {

fprintf(stderr,"Document not parsed successfully. ＼n");

return;

}

⑤cur = xmlDocGetRootElement(doc);

⑥if (cur == NULL) {

fprintf(stderr,"empty document＼n");

xmlFreeDoc(doc);

return;

}

⑦if (xmlStrcmp(cur->name, (const xmlChar \*) "story")) {

fprintf(stderr,"document of the wrong type, root node != story");

xmlFreeDoc(doc);

return;

}

①定义解析文件指针。

②定义结点指针(你需要他为了在各个结点间移动)。

④检查解析文件是否成功，如果不成功，libxml将指一个注册的错误并停止。

注释

一个常见错误是不适当的编码。XML标准文件除了用UTF-8或UTF-16外还可用其他编码保存。如果文件是这样，libxml将自动地为你转换到UTF-8。更多关于XML编码信息包含在XML标准中。

⑤取得文件根元素

⑥检查确认当前文件中包含内容。

⑦在这个例子中，我们需要确认文件是正确的类型。“Story”是在这个指南中使用文件的根类型。

取得元素内容

你找到在文件树中你要查找的元素后能取得他的内容。在这个例子中我们查找“story”元素。进程将在冗长的树中查找我们感兴趣的元素。我们假定期你已有了一个名为doc的xmlDocPtr和一个名为cur的xmlNodPtr。

①cur = cur->xmlChildrenNode;

②while (cur != NULL) {

if ((!xmlStrcmp(cur->name, (const xmlChar \*)"storyinfo"))){

parseStory (doc, cur);

}

cur = cur->next;

}

①取得cur的第一个子结点，cur指向文件的根，即“story”元素。

②这个循环迭代通过“story”的子元素查找“storyinfo”。这是个包含有我们将查找的“keywords”的元素。他使用了libxml字符串比较函数xmlStrcmp。如果相符，他调用函数parseStory。

void

parseStory (xmlDocPtr doc, xmlNodePtr cur) {

xmlChar \*key;

① cur = cur->xmlChildrenNode;

②while (cur != NULL) {

if ((!xmlStrcmp(cur->name, (const xmlChar \*)"keyword"))) {

③ key = xmlNodeListGetString(doc, cur->xmlChildrenNode, 1);

printf("keyword: %s＼n", key);

xmlFree(key);

}

cur = cur->next;

}

return;

}

① 再次取得第一个子结点。

② 像上面那个循环相同，我们能过迭代，查找我们感兴趣的叫做“keyword”的元素。

③ 当我们找到元素“keyword”时，我们需要打印他包含在XML中的记录的内容，文本被包含于元素的子结点中，因此我们借助了cur->xmlChildrenNode，为了取得文本，我们使用函数xmlNodeListGetString，他有一个文件指针参数，在这个例子中，我们仅仅打印他。

注释

因为xmlNodeListGetString为他返回的字符串分配内存，你必须使用xmlFree释放他。

使用XPath取得元素内容

除了一步步遍历文件树查找元素外，Libxml2包含支持使用Xpath表达式取得指定结点集。完整的Xpath API文件在这里。Xpath允许通过路径文件搜索匹配指定条件的结点。在下面的例子中，我们搜索文件中所有的“keyword”元素。

注释

下面是Xpath完整的讨论。他周详的使用资料，请查阅Xpath规范。

这个例子完整的代码参见附录D，XPath例程代码。

Using XPath requires setting up an xmlXPathContext and then supplying the XPath expression and the context to the xmlXPathEvalExpression

function.

The function returns an xmlXPathObjectPtr, which includes the set of nodes satisfying the XPath expression.

使用XPath需要安装xmlXPathContext才支持XPath表达式及xmlXPathEvalExpression函数，这个函数返回一个xmlXPathObjectPtr，他包含有

XPath表达式的结点集。

xmlXPathObjectPtr

getnodeset (xmlDocPtr doc, xmlChar \*xpath){

①xmlXPathContextPtr context;

xmlXPathObjectPtr result;

②context = xmlXPathNewContext(doc);

③result = xmlXPathEvalExpression(xpath, context);

④if(xmlXPathNodeSetIsEmpty(result->nodesetval)){

printf("No result＼n");

return NULL;

}

xmlXPathFreeContext(context);

return result;

}

①首先定义变量

②初始化变量context

③应用XPath表达式

④检查结果

由函数返回的xmlPathObjectPtr包含一个结点集和其他需要被迭代及操作的信息。在这个例子中我们的函数返回xmlXPathObjectPtr，我们使用他打印我们文件中keyword结点的内容。这个结点集对象包含在集合(nodeNr)中的元素数目及一个结点(nodeTab)数组。

①for (i=0; i nodeNr; i++) {

②keyword = xmlNodeListGetString(doc,

nodeset->nodeTab->xmlChildrenNode, printf("keyword: %s＼n", keyword);

xmlFree(keyword);

}

①变量nodeset->Nr持有结点集中元素的数量。我们使用他遍历数组。

②打印每个结点包含的内容。

注释

Note that we are printing the child node of the node that is returned, because the contents of the keyword element are a child text node.注意我们打印的是结点的子结点的返回值，因为keyword元素的内容是个子文本结点。

写元素

写元素内容使用上面许多相同的步骤?解析文件并遍历树。我们先解析文件然后遍历树查找我们想插入元素的位置。在这个例子中，我们再一次查找“storyinfo

”元素并插入一个keyword。然后我们装文件写入磁盘。完整代码：附录E，添加keyword例程

本例中主要的不同在于parseStory

void

parseStory (xmlDocPtr doc, xmlNodePtr cur, char \*keyword) {

①xmlNewTextChild (cur, NULL, "keyword", keyword);

return;

}

①XmlNewTextChild函数添加一个当前结点的新的子元素到树中

一旦结点被添加，我们应当写文件到文件中。你是否想给元素指定一个命名空间？你能添加他，在我们的例子中，命名空间是NULL。

xmlSaveFormatFile (docname, doc, 1);

第一个参数是写入文件的名，你注意到和我们刚刚读入的文件名是相同的。在这个例子中，我们仅仅覆盖原来的文件。第二个参数是个xmlDoc结构指针，第三个参数设定为1，确保在输出上写入。

libxml（二）

写属性

写属性类似于给一个新元素写文本。在这个例子中，我们将添加一个reference结点URI属性到我们的文件中。完整代码：附录F，添加属性例程代码。reference是story元素的一个子结点，所以找到并插入新元素及其属性是简单的。一旦我们在parseDoc进行了错误检查，我们将在正确的位置加放我们的新元素。但进行之前我们需要定义一个此前我们不见过的数据类型。

xmlAttrPtr newattr;

我们也需要xmlNodePtr：

xmlNodePtr newnode;

剩下的parseDoc则和前面相同，检查根结点是否为story。如果是的，那我们知道我们将在指定的位置添加我们的元素。

① newnode = xmlNewTextChild (cur, NULL, "reference", NULL);

②newattr = xmlNewProp (newnode, "uri", uri);

①使用xmlNewTextChild函数添国一个新结点到当前结点位置。

一旦结点被添加，文件应像前面的例子将我们添加的元素及文本内容写入磁盘。

取得属性

取得属性值类似于前面我们取得一个结点的文本内容。在这个例子中，我们将取出我们在前一部分添加的URI的值。完整代码：附录G，取得属性值例程代码。

   这个例子的初始步骤和前面是类似的：解析文件，查找你感兴趣的元素，然后进入一个函数完成指定的请求任务。在这个例子中，我们调用getReference。

void

getReference (xmlDocPtr doc, xmlNodePtr cur) {

xmlChar \*uri;

cur = cur->xmlChildrenNode;

while (cur != NULL) {

if ((!xmlStrcmp(cur->name, (const xmlChar \*)"reference"))) {

① uri = xmlGetProp(cur, "uri");

printf("uri: %s＼n", uri);

xmlFree(uri);

}

cur = cur->next;

}

return;

}

① 关键函数是xmlGetProp,他返回一个包含属性值的xmlChar。在本例中，我们仅仅打印他。

注释

如果你使用DTD定义属性的固定值或缺省值，这个函数也将取得他。

编码转换

数据编码兼容问题是程式员新建普通的XML或特定XML时最常见的困难。按照这里

   稍后的讨论来思考设计你的应用程式将帮助你避免这个困难。实际上，libxml能以UTF-8格式保存和操纵多种数据

你的程式使用其他的数据格式，比如常见的ISO-8859-1编码，必须使用libxml函数转换到UTF-8。如果你想你的程式以除UTF-8外的其他编码方式输出也必须做转换。

   如果能有效地转换数据Libxml将使用转换器。无转换器时，仅仅UTF-8、UTF-16和ISO-8859-1能够被作为外部格式使用。有转换器时，他能将从其他格式和UTF-8互换的所有格式均可使用。当前转换器支持大约150种不同的编码格式之间的相互转换。实际支持的格式数量正在被实现。每一个实目前的转换器尽可能的支持每一种格式。

警告

一个常见错误是在内部数据不同的部分使用不同的编码格式。最常见的是情况是个应用以ISO-8859-1作为内部数据格式，结合libxml部分使用UTF-8格式。结果是个应用程式要面对不同地内部数据格式。一部分代码执行后，他或其他部分代码将使用曲解的数据。

   这个例子构造一个简单的文件，然后添加在命令行提供的内容到根元素并使用适当的编码将结果输出到标准输出设备上。在这个例子中，我们使用ISO-8859-1编码。在命令输入的内容将被从ISO-8859-1转换到UTF-8。完整代码：附件H，编码转换例程代码。

   包含在例子中的转换函数使用libxml的xmlFindCharEncodingHandler函数。

①xmlCharEncodingHandlerPtr handler;

②size = (int)strlen(in)+1;

out\_size = size\*2-1;

out = malloc((size\_t)out\_size);

…

③handler = xmlFindCharEncodingHandler(encoding);

…

④handler->input(out, &out\_size, in, &temp);

…

⑤xmlSaveFormatFileEnc("-", doc, encoding, 1);

①定义一个xmlCharEncodingHandler函数指针。

②XmlCharEncodingHandler函数需要给出输入和输出字符串的大小，这里计算输入输出字符串。

③XmlFindCharEncodingHandler使用数据初始编码作为参数搜索libxml已完成的转换器句柄并将找到的函数指针返回，如果没有找到则返回NULL。

④The conversion function identified by handler requires as its arguments pointers to the input and output strings, along with the length of each. The lengths must be determined separately by the application.

由句柄指定的转换函数请求输入、输出字符中及他们的长度作为参数。这个长度必须由应用程式分别指定。

⑤用指定编码而不是UTF-8输出，我们使用xmlSaveFormatFileEnc指不定期编码方式。

A. 编译

Libxml包含一个脚本xml2-config，他一般用于编译和链接程式到库时产生标志。

  为了取得预处理和编译标志，使用xml2-config ?cflags，为了取得链接标志，使用xml2-config ?libs。其他有效的参数请使用xml2-config ?help查阅。

B. 示例文件

John Fleck

June 2, 2002

example keyword

This is the headline

This is the body text.

C. Keyword例程代码

#include

#include

#include

#include

#include

void

parseStory (xmlDocPtr doc, xmlNodePtr cur) {

xmlChar \*key;

cur = cur->xmlChildrenNode;

while (cur != NULL) {

if ((!xmlStrcmp(cur->name, (const xmlChar \*)"keyword"))) {

key = xmlNodeListGetString(doc, cur->xmlChildrenNode, 1);

printf("keyword: %s＼n", key);

xmlFree(key);

}

cur = cur->next;

}

return;

}

static void

parseDoc(char \*docname) {

xmlDocPtr doc;

xmlNodePtr cur;

doc = xmlParseFile(docname);

if (doc == NULL ) {

fprintf(stderr,"Document not parsed successfully. ＼n");

return;

}

cur = xmlDocGetRootElement(doc);

if (cur == NULL) {

fprintf(stderr,"empty document＼n");

xmlFreeDoc(doc);

return;

}

if (xmlStrcmp(cur->name, (const xmlChar \*) "story")) {

fprintf(stderr,"document of the wrong type, root node != story");

xmlFreeDoc(doc);

return;

}

cur = cur->xmlChildrenNode;

while (cur != NULL) {

if ((!xmlStrcmp(cur->name, (const xmlChar \*)"storyinfo"))){

parseStory (doc, cur);

}

cur = cur->next;

}

xmlFreeDoc(doc);

return;

}

int

main(int argc, char \*\*argv) {

char \*docname;

if (argc

libxml（三）

D. XPath例程代码

#include

#include

xmlDocPtr

getdoc (char \*docname) {

xmlDocPtr doc;

doc = xmlParseFile(docname);

if (doc == NULL ) {

fprintf(stderr,"Document not parsed successfully. ＼n");

return NULL;

}

return doc;

}

xmlXPathObjectPtr

getnodeset (xmlDocPtr doc, xmlChar \*xpath){

xmlXPathContextPtr context;

xmlXPathObjectPtr result;

context = xmlXPathNewContext(doc);

result = xmlXPathEvalExpression(xpath, context);

if(xmlXPathNodeSetIsEmpty(result->nodesetval)){

printf("No result＼n");

return NULL;

}

xmlXPathFreeContext(context);

return result;

}

int

main(int argc, char \*\*argv) {

char \*docname;

xmlDocPtr doc;

xmlChar \*xpath = ("//keyword");

xmlNodeSetPtr nodeset;

xmlXPathObjectPtr result;

int i;

xmlChar \*keyword;

if (argc nodesetval;

for (i=0; i nodeNr; i++) {

keyword = xmlNodeListGetString(doc, nodeset->nodeTab->printf

("keyword: %s＼n", keyword);

xmlFree(keyword);

}

xmlXPathFreeObject (result);

}

xmlFreeDoc(doc);

xmlCleanupParser();

return (1);

}

E. 添加keyword例程代码

#include

#include

#include

#include

#include

void

parseStory (xmlDocPtr doc, xmlNodePtr cur, char \*keyword) {

xmlNewTextChild (cur, NULL, "keyword", keyword);

return;

}

xmlDocPtr

parseDoc(char \*docname, char \*keyword) {

xmlDocPtr doc;

xmlNodePtr cur;

doc = xmlParseFile(docname);

if (doc == NULL ) {

fprintf(stderr,"Document not parsed successfully. ＼n");

return (NULL);

}

cur = xmlDocGetRootElement(doc);

if (cur == NULL) {

fprintf(stderr,"empty document＼n");

xmlFreeDoc(doc);

return (NULL);

}

if (xmlStrcmp(cur->name, (const xmlChar \*) "story")) {

fprintf(stderr,"document of the wrong type, root node != story");

xmlFreeDoc(doc);

return (NULL);

}

cur = cur->xmlChildrenNode;

while (cur != NULL) {

if ((!xmlStrcmp(cur->name, (const xmlChar \*)"storyinfo"))){

parseStory (doc, cur, keyword);

}

cur = cur->next;

}

return(doc);

}

int

main(int argc, char \*\*argv) {

char \*docname;

char \*keyword;

xmlDocPtr doc;

if (argc

#include

#include

#include

#include

xmlDocPtr

parseDoc(char \*docname, char \*uri) {

xmlDocPtr doc;

xmlNodePtr cur;

xmlNodePtr newnode;

xmlAttrPtr newattr;

doc = xmlParseFile(docname);

if (doc == NULL ) {

fprintf(stderr,"Document not parsed successfully. ＼n");

return (NULL);

}

cur = xmlDocGetRootElement(doc);

if (cur == NULL) {

fprintf(stderr,"empty document＼n");

xmlFreeDoc(doc);

return (NULL);

}

if (xmlStrcmp(cur->name, (const xmlChar \*) "story")) {

fprintf(stderr,"document of the wrong type, root node != story");

xmlFreeDoc(doc);

return (NULL);

}

newnode = xmlNewTextChild (cur, NULL, "reference", NULL);

newattr = xmlNewProp (newnode, "uri", uri);

return(doc);

}

int

main(int argc, char \*\*argv) {

char \*docname;

char \*uri;

xmlDocPtr doc;

if (argc

#include

#include

#include

#include

void

getReference (xmlDocPtr doc, xmlNodePtr cur) {

xmlChar \*uri;

cur = cur->xmlChildrenNode;

while (cur != NULL) {

if ((!xmlStrcmp(cur->name, (const xmlChar \*)"reference"))) {

uri = xmlGetProp(cur, "uri");

printf("uri: %s＼n", uri);

xmlFree(uri);

}

cur = cur->next;

}

return;

}

void

parseDoc(char \*docname) {

xmlDocPtr doc;

xmlNodePtr cur;

doc = xmlParseFile(docname);

if (doc == NULL ) {

fprintf(stderr,"Document not parsed successfully. ＼n");

return;

}

cur = xmlDocGetRootElement(doc);

if (cur == NULL) {

fprintf(stderr,"empty document＼n");

xmlFreeDoc(doc);

return;

}

if (xmlStrcmp(cur->name, (const xmlChar \*) "story")) {

fprintf(stderr,"document of the wrong type, root node != story");

xmlFreeDoc(doc);

return;

}

getReference (doc, cur);

xmlFreeDoc(doc);

return;

}

int

main(int argc, char \*\*argv) {

char \*docname;

if (argc

#include

unsigned char\*

convert (unsigned char \*in, char \*encoding)

{

unsigned char \*out;

int ret,size,out\_size,temp;

xmlCharEncodingHandlerPtr handler;

size = (int)strlen(in)+1;

out\_size = size\*2-1;

out = malloc((size\_t)out\_size);

if (out) {

handler = xmlFindCharEncodingHandler(encoding);

if (!handler) {

free(out);

out = NULL;

}

}

if (out) {

temp=size-1;

ret = handler->input(out, &out\_size, in, &temp);

if (ret || temp-size+1) {

if (ret) {

printf("conversion wasn’t successful.＼n");

} else {

printf("conversion wasn’t successful. converted: }

free(out);

out = NULL;

} else {

out = realloc(out,out\_size+1);

out[out\_size]=0; /\*null terminating out\*/

}

} else {

printf("no mem＼n");

}

return (out);

}

int

main(int argc, char \*\*argv) {

unsigned char \*content, \*out;

xmlDocPtr doc;

xmlNodePtr rootnode;

char \*encoding = "ISO-8859-1";

if (argc

char \*convert(char \*instr,char \*encoding)

{

xmlCharEncodingHandlerPtr handler;

xmlBufferPtr in,out;

handler = xmlFindCharEncodingHandler(encoding);

if(NULL != handler)

{

in = xmlBufferCreate();

xmlBufferWriteChar(in,instr);

out = xmlBufferCreate();

if(xmlCharEncInFunc(handler, out, in) 〈 0)

{

xmlBufferFree(in);

xmlBufferFree(out);

return NULL;

}

else

{

xmlBufferFree(in);

return (char \*)out-〉content;

}

}

}