译前言

翻译本文缘起我们有一个项目使用了 libxml，却出现内存泄漏的情况。于是我想了解一点 libxml 的知识，却苦于没有入门的中文材料。偶尔发现网上有人翻译了这个教程，却不是很合自己用，便想自己翻译一下。

只是自己中英文皆不精，XML 也不熟，中间有错误自己都看不出来。所以本文仅供初接触 libxml 的，又懒得看原文的人士看一下，别的用处想必不大。

英文版本见http://www.xmlsoft.org/tutorial/index.html，本文仅仅是对英文版的简单翻译。若有疑问，请直接查看英文原处。

2006-06-16

Libxml 简单教程

John Fleck

目录

摘要

Libxml 是一个自由授权的，用来处理 XML 的 C 语言库，它可以移植到很多的平台上。本教程为其基本函数提供了例子。

介绍

Libxml 是一个 C 语言库，用来对 XML 数据进行读取、创建和维护。这个教程对其基本功能提供了例子代码和说明。

在 libxml 的项目主页 http://www.xmlsoft.org/上提供了关于它如何使用的更加详尽的信息，包括完整的 API 文档 http://xmlsoft.org/html/libxml-lib.html。本教程不能替代这些文档，而仅仅用来说明用这个库来实现 XML 基本操作时所用到的函数。

本教程中用例子代码来演示下列内容：

- 解析文档；
- 取出指定元素里的文本；
- 添加元素以及它的内容；
- 添加属性；
- 取出属性的值。

全部例子代码见附录。

数据类型

Libxml 声明了一些数据类型，用来隐藏那些除非有特别需要就不用去理会的复杂细节，这些数据类型我们总是会一次一次的遇到。

xmlChar

对char的基本代替，是一个UTF-8 编码字符串中的一个字节。如果你的数据使用了其他编码，在使用libxml函数前就必须转换为UTF-8。关于编码的更多信息见http://www.xmlsoft.org/encoding.html。

xmlDoc 和 xmlDocPtr

是一个包含了从解析文档后创建出的树的结构。xmlDocPtr 是指向该结构的指针。

xmlNode 和 xmlNodePtr

包含单个节点的结构。xmlNodePtr 是指向该结构的指针，它用来遍历文档树。

解析文件

解析文件只需要文件的名字和简单的一个函数调用，再加上错误检查就可以了。完整代码见 附录 C Keyword 例子代码

```
① xmlDocPtr doc;
② xmlNodePtr cur;

③ doc = xmlParseFile(docname);

④ if (doc == NULL ) {
        fprintf(stderr,"Document not parsed successfully. \n");
        return;
}

⑤ cur = xmlDocGetRootElement(doc);

⑥ if (cur == NULL) {
        fprintf(stderr,"empty document\n");
        xmlFreeDoc(doc);
        return;
}

⑦ if (xmlStrcmp(cur->name, (const xmlChar *) "story")) {
        fprintf(stderr,"document of the wrong type, root node != story");
        xmlFreeDoc(doc);
        return;
}
```

① 声明指向你要解析的文档的指针。
② 声明一个节点指针（处理单个的节点的时候需要它）。
④ 检查解析文档是否成功。如果没有，libxml 会在此处报错并终止。
→ 注意
此处常见的错误是对编码的错误处理。XML标准要求，如果一个文档不是用UTF-8 或UTF-16 编码的，文档内必须包含有编码的明确声明。如果文档声明了编码，libxml会自动的为你做必要的到UTF-8的编码转换。关于XML编码要求的更多详情见http://www.w3.org/TR/REC-xml#charencoding。
⑤ 获取文档的根元素。
⑥ 检查确认文档包含了东西。
⑦ 在我们的例子里，我们需要确认文档内容正确，”story”是在本教程中使用到的文档的根类型。

获取元素内容

获取一个元素的内容需要遍历文档树直到发现你要找的东西。下面的例子，我们要找一个叫”keyword”的元素，它保存在一个叫”storyinfo”的元素里面。寻找这个节点的过程就需要遍历这棵树，这比较令人郁闷。我们假设你已经有了一个叫 doc 的 xmlDocPtr 和叫 cur 的 xmlNodPtr 变量。

```
① cur = cur->xmlChildrenNode;
② while (cur != NULL) {
        if ((!xmlStrcmp(cur->name, (const xmlChar *)"storyinfo"))){
                parseStory (doc, cur);
        }

cur = cur->next;
}
```

① 获得 cur 的第一个子节点。此时，cur 指向文档的根，也就是”story”元素。
② 这个循环遍历”story 的所有子元素，寻找叫”storyinfo”的元素，也就是那个我们要寻找的包含”keyword”的那个元素。这里使用了 libxml 的字符串比较函数--xmlStrcmp。如果有匹配的，就调用函数 parseStory。

```
void
parseStory (xmlDocPtr doc, xmlNodePtr cur) {

    xmlChar *key;
    ① cur = cur->xmlChildrenNode;
    ② while (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 又是获得了第一个子节点。
② 就像上个循环，我们遍历节点，寻找感兴趣的那个。在这个例子里是”keyword”。
③ 我们发现”keyword”元素之后，就打印出来。要记住，在 XML 中，一个元素包含的文本是这个元素的一个子节点，所以我们转向 cur->xmlChildrenNode。要获取它，我们使用函数 xmlNodeListGetString，它也要把 doc 指针作为一个参数。在这儿，我们仅仅把它打印出来。

→ 注意

因为 xmlNodeListGetString 为它返回的字符串分配了动态内存，你必须用 xmlFree 来释放它。

使用 Xpath 来获取元素内容

除了用遍历文档树的方式来寻找一个元素外，libxml2 还支持使用XPath表达式来获取满足指定条件的节点集合。关于XPath API 的完整文件见http://xmlsoft.org/html/libxml-xpath.html。

XPath 允许在一个文档中寻找到符合条件的所有节点。下面的例子中，我们在文档中寻找到所有的”keyword”元素。

→ 注意

对XPath 的完整讨论超出了本文的范畴，关于它使用的详情参见http://www.w3.org/TR/xpath。

本例子的完整代码在 附录 D XPath 例子程序。

使用 XPath 就要先设置一个 xmlXPathContext，然后把这个 XPath 表达式和上下文提交给 xmlXPathEvalExpression 函数。函数返回 xmlXPathObjectPtr，它包含着符合 XPath 表达式的节点集合。

```
xmlXPathObjectPtr
getnodeset (xmlDocPtr doc, xmlChar *xpath){

① xmlXPathContextPtr context;
xmlXPathObjectPtr result;

② context = xmlXPathNewContext(doc);
③ result = xmlXPathEvalExpression(xpath, context);
④ if(xmlXPathNodeSetIsEmpty(result->nodesetval)){
        xmlXPathFreeObject(result);
         printf("No result\n");
        return NULL;
}
```

① 首先我们声明变量。
② 初始化 context 变量。
③ 应用 XPath 表达式。
④ 检查结果，如果没有结果，释放分配给结果的动态内存。

函数返回的 xmlPathObjectPtr 包含着节点集合以及用来遍历集合和操作结果的所需要的其它信息。在本例中，我们的函数返回了 xmlXPathObjectPtr。我们用它来打印我们文档中 keyword 节点的内容。节点集合对象包含了集合中元素的个数(nodeNr)和一个节点的数组(nodeTab)。

```
① for (i=0; i < nodeset->nodeNr; i++) {
② keyword = xmlNodeListGetString(doc, nodeset->nodeTab[i]->xmlChildrenNode, 1);
        printf("keyword: %s\n", keyword);
         xmlFree(keyword);
}
```

① nodeset->Nr 保存着节点集合中元素的个数。这里我们用来遍历数组。
② 在这儿我们打印每个返回的节点的内容。

→ 注意

注意我们打印的是返回的节点的子节点，因为”keyword”元素的内容是个子文本节点。

写入元素内容

写入元素内容的步骤和上面我们所用到的差不多—都是解析文档然后遍历树。我们解析文档，然后遍历树寻找我们想要插入元素的地方。在下面的例子里，我们还是寻找”storyinfo”元素，但是我们这次插入一个 keyword。然后我们把文件写回磁盘。完整代码见 附录 E 增加 keyword 例子代码。

本例子中最主要的差别在 parseStory 函数中：

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

        ① xmlNewTextChild (cur, NULL, "keyword", keyword);
    return;
}
```

① xmlNewTextChil 函数在由 cur 指定的当前节点指针的位置添加了一个新子元素。

一旦这个节点添加成功，我们就可以把文档写回文件。如果你想让该元素有一个 namespace，也是在这儿添加。在本例中，namespace 是 NULL。

```
        xmlSaveFormatFile (docname, doc, 1);
```

第一个参数是要写入的文件名。你会注意到它和我们刚刚读的文件名相同。在本例中，我们把老文件给覆盖写了。第二个参数指向 xmlDoc 结构。第三个参数设为 1 保证输出会缩格。

写入属性

写入属性和向一个新元素中写入文本及其类似。在本例中，我们向我们文档中增加一个 reference URI。文件代码见 附录 F 增加属性例子代码。

reference 是 story 元素的子节点。所以找到该位置来添加我们的新元素和属性非常的简单。在 parseDoc 中一进行完错误检查，就正是添加我们的元素的地方。但在去这样做之前，我们需要声明一个我们还没有见过的数据类型的变量。

```
xmlAttrPtr newattr;
```

我们还需要另外一个 xmlNodePtr：

```
xmlNodePtr newnode;
```

parseDoc 函数在检查根元素是否是 story 之前的部分和原来完全相同。如果是 story，我们就知道这就是添加我们的元素的地方。

```
① newnode = xmlNewTextChild (cur, NULL, "reference", NULL);
② newattr = xmlNewProp (newnode, "uri", uri);
```

① 首先，我们当前节点指针 cur 所指的位置上添加一个新节点。使用 xmlNewTextChild 函数。

添加完节点后，和前面增加一个文本内容的元素的例子一样，把文件写回磁盘。

获取属性

获取一个属性的值和我们前面获取一个节点的文本内容的例子非常的类似。在本例中，我们会获取在上节中添加的 URI 的值。完整代码见：附录 G 获取属性值例子代码

本例的基本步骤和前面的非常的类似：解析文档，寻找你感兴趣的元素，调用相应函数来完成所要求的任务。在本例中，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 xmlGetProp，它返回一个保存着属性值的 xmlChar，在本例中，我们仅仅把它打印出来。

→ 注意

如果你用 DTD 为该属性声明了一个固定或缺省的属性，这个函数会获取到该值。

编码转换

数据编码兼容问题是开发者在初次使用 XML 和 libxml 的时候最常遇到的问题。在你设计应用的时候就要把这个问题想清楚，免得将来更加麻烦。libxml 在内部是用 UTF-8 格式来储存和维护数据的。你程序中用到的其他格式数据，例如常用的 ISO-8859-1 编码，必须在传给 libxml 函数之前转换为 UTF-8。如果你想你的程序的输出不是 UTF-8 的编码，同样也需要转换。

如果有 iconv，libxml 就可以用它来转换数据。如果没有 iconv，数据的外部格式只能使用 UTF-8、UTF-16 以及 ISO-8859-1。但有了 iconv，只要能够用 iconv 和 UTF-8 来回转换的编码格式就都可以使用了。现在 iconv 可以支持大约 150 种可以来回转换的字符格式。虽然每个 iconv 实现实际支持格式数量不同，但几乎每个实现都能保证能够支持我们平常所遇到的格式。

△！ 警告

一个常见的错误是在某个代码的不同部分内部数据使用了不同的格式。常见的例子是：一个应用采用 ISO-8859-1 作为内部数据格式，来和 libxml 联编；libxml 却是用 UTF-8 作为内部数据格式的。结果就是一个应用内，执行的不同代码段把内部数据视为不同的格式。自然的，其中总会有几个代码段会把数据解释错。

本例中，构造了一个简单文档，然后用正确的编码把命令行输入的内容添加到文档的根元素，然后把结果输出到标准输出。本例中，我们使用 ISO-8859-1 编码，命令行输入的字符串的编码从 ISO-8859-1 到 UTF-8。完整编码见 附录 H 编码转换例子代码

转换器，封装成了例子代码中的转换函数，使用了 libxml 的 xmlFindCharEncodingHandler 函数来获取。

```
① xmlCharEncodingHandlerPtr handler;
② size = (int)strlen(in)+1;
out_size = size*2-1;
out = malloc((size_t)out_size);

…
③ handler = xmlFindCharEncodingHandler(encoding);
…
④ handler->input(out, &out_size, in, &temp);
…
⑤ xmlSaveFormatFileEnc("-", doc, encoding, 1);
```

① 声明一个处理器指针，指向 xmlCharEncodingHandler 函数。
② xmlCharEncodingHandler 函数需要知道输入和输出的字符串的长度，于是这儿计算了输入和输出的字符串的长度。
③ xmlFindCharEncodingHandler 把数据的原始编码作为参数，在 libxml 内置的转换处理器中查找。返回一个函数的指针，没有找到就返回 NULL。
④ 处理器所标示的转换函数需要输入和输出的字符串的指针以及它们的长度作为参数。两个长度都必须由应用来确定。
⑤ 用不是 UTF-8 的编码来输出，我们需要使用 xmlSaveFormatFileEnc，并要指定编码。

A. 编译

Libxml包含了一个脚本，xml2-conf，用来产生基于该库编写的程序的编译和连接标志。对预处理器和编译器标志，使用**xml2-config －cflags**。对库连接标志，使用**xml2-config －libs**。其他选项使用**xml2-config －help**查看。

B. Sample Document

```
<?xml version="1.0"?>
<story>
  <storyinfo>
    <author>John Fleck</author>
    <datewritten>June 2, 2002</datewritten>
    <keyword>example keyword</keyword>
  </storyinfo>
  <body>
    <headline>This is the headline</headline>
    <para>This is the body text.</para>
  </body>
</story>
```

C. Code for Keyword Example

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <libxml/xmlmemory.h>
#include <libxml/parser.h>

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 <= 1) {
                printf("Usage: %s docname\n", argv[0]);
                return(0);
        }

        docname = argv[1];
        parseDoc (docname);
```

```
        return (1);
}
```

D. Code for XPath Example

```
#include <libxml/parser.h>
#include <libxml/xpath.h>

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);
        if (context == NULL) {
                printf("Error in xmlXPathNewContext\n");
                return NULL;
        }
        result = xmlXPathEvalExpression(xpath, context);
        xmlXPathFreeContext(context);
        if (result == NULL) {
                printf("Error in xmlXPathEvalExpression\n");
                return NULL;
        }
        if(xmlXPathNodeSetIsEmpty(result->nodesetval)){
                xmlXPathFreeObject(result);
                 printf("No result\n");
                return NULL;
        }
        return result;
}
int
main(int argc, char **argv) {
```

```
        char *docname;
        xmlDocPtr doc;
        xmlChar *xpath = (xmlChar*) "//keyword";
        xmlNodeSetPtr nodeset;
        xmlXPathObjectPtr result;
        int i;
        xmlChar *keyword;

        if (argc <= 1) {
                printf("Usage: %s docname\n", argv[0]);
                return(0);
        }

        docname = argv[1];
        doc = getdoc(docname);
        result = getnodeset (doc, xpath);
        if (result) {
                nodeset = result->nodesetval;
                for (i=0; i < nodeset->nodeNr; i++) {
                        keyword = xmlNodeListGetString(doc,
nodeset->nodeTab[i]->xmlChildrenNode, 1);
                printf("keyword: %s\n", keyword);
                xmlFree(keyword);
                }
                xmlXPathFreeObject (result);
        }
        xmlFreeDoc(doc);
        xmlCleanupParser();
        return (1);
}
```

E. Code for Add Keyword Example

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <libxml/xmlmemory.h>
#include <libxml/parser.h>

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 <= 2) {
                printf("Usage: %s docname, keyword\n", argv[0]);
                return(0);
        }

        docname = argv[1];
        keyword = argv[2];
        doc = parseDoc (docname, keyword);
        if (doc != NULL) {
```

```
        xmlSaveFormatFile (docname, doc, 0);
        xmlFreeDoc(doc);
    }

    return (1);
}
```

F. Code for Add Attribute Example

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <libxml/xmlmemory.h>
#include <libxml/parser.h>


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 <= 2) {
                printf("Usage: %s docname, uri\n", argv[0]);
                return(0);
        }

        docname = argv[1];
        uri = argv[2];
        doc = parseDoc (docname, uri);
        if (doc != NULL) {
                xmlSaveFormatFile (docname, doc, 1);
                xmlFreeDoc(doc);
        }
        return (1);
}
```

G. Code for Retrieving Attribute Value Example

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <libxml/xmlmemory.h>
#include <libxml/parser.h>

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 <= 1) {
                printf("Usage: %s docname\n", argv[0]);
                return(0);
        }

        docname = argv[1];
        parseDoc (docname);

        return (1);
}
```

H. Code for Encoding Conversion Example

```
#include <string.h>
#include <libxml/parser.h>
```

```
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: %i
octets.\n",temp);
                        }
                        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 <= 1) {
                printf("Usage: %s content\n", argv[0]);
                return(0);
        }

        content = argv[1];

        out = convert(content, encoding);

        doc = xmlNewDoc ("1.0");
        rootnode = xmlNewDocNode(doc, NULL, (const xmlChar*)"root", out);
        xmlDocSetRootElement(doc, rootnode);

        xmlSaveFormatFileEnc("-", doc, encoding, 1);
        return (1);
}
```

I. 感谢

很多朋友为本教程慷慨的提供了反馈、代码以及改进建议。他们是（名字不分先后）：Daniel Veillard, Marcus Labib Iskander, Christopher R. Harris, Igor Zlatkovic, Niraj Tolia, David Turover