Cómo documentar código fuente en C con cxref.

Cxref es una utilidad para documentar código escrito en C a partir de los comentarios incluidos en los archivos fuentes, generando además referencias cruzadas de las funciones y variables utilizadas en el programa. Puede producir la documentación en los formatos LaTeX, HTML, SGML o RTF.
Las referencias cruzadas incluyen, para cada función:

-El nombre del archivo donde está su prototipo.
-Funciones llamadas y sus archivos.
-Funciones que la llaman y sus archivos.
-Variables globales utilizadas y sus archivos.

Es decir que, cuando consultamos en la documentación generada una determinada función, podemos saber en qué archivo está su prototipo, qué funciones la llaman y en qué archivos están, y qué funciones son llamadas desde ella y en qué archivos están. (Excepto las de bibliotecas que no pertenezcan al programa, de las que sólo muestra los nombres), además de poder leer el comentario que el desarrollador quiso escribir para esa función o variable.

Su modo de uso es bastante sencillo, se trata de configurar de forma especial los comentarios del C:
Para incluir documentación referente al archivo, se han de añadir, preferiblemente al principio, caracteres '*' extra después y antes, respectivamente, de los caraceres de principio y fin de comentario en C:

```
/** Este archivo tiene que estar escrito en C para que cxref funcione. **/
```

Para incluir comentarios referentes a las variables y funciones, se han de añadir caracteres '+' extra, situando el comentario inmediatamente antes del nombre de la función o variable, análogamente a como antes se añadieron asteriscos:

```
/*+ Esto es un comentario para una función o variable que cxref entenderá. +*/
void funcion_con_comentario (void) { printf ("Esto no hace nada.\n"); }
```

Puede incluirse cualquier número de caracteres '*' o '+' entre los delimitadores de comentarios estándares del C: "/*" y "*/"

cxref colocará el comentario antes de los **#defines**, y después del nombre de variable o función.
La descripción de la función, con sus parámetros y el tipo devuelto, se insertará en el lugar correspondiente a la primera línea en blanco que haya en el comentario, o al final del mismo.

Son detalles a tener en cuenta los siguientes:

Cómo documentar código fuente en C con cxref.

-Cada archivo .h tiene que tener todos los #includes necesarios. A cxref no le vale que nos ahorremos un #include en un archivo porque ya esté en otro anterior, es decir:

Archivo uno.h:
```
#include "cero.h"
```

Archivo dos.h:
```
#include "uno.h"
```

Si en el archivo dos.h se utilizan variables o funciones del archivo cero.h, hay que incluirlo expresamente también:

Archivo dos.h:
```
#include "cero.h"
#include "uno.h"
```

Los comentarios a las funciones deben estar en los archivos .c, no en los .h.
A continuación, un ejemplo de la salidad generada a partir de un archivo .c:

```
File src/xmlmisc.c
Miscelánea de funciones que utilizan libxml.

Included Files

    * #include <string.h>

    * #include "src/gtkmisc.h"
          o #include <gtk/gtk.h>

    * #include "src/xmlmisc.h"
          o #include <libxml/xmlmemory.h>
          o #include <libxml/parser.h>
          o #include <libxml/tree.h>

Global Function nuevo_archivo_xml()
Recibe el nombre del archivo de disco a crear y el de su nodo raíz. Devuelve un
xmlDocPtr con el archivo creado.

xmlDocPtr nuevo_archivo_xml ( char* nomArchivo, char* nomNodoRaiz )
Prototyped in:    src/xmlmisc.h
Calls:      xmlDocSetRootElement(), xmlNewDoc(), xmlNewDocNode()
Called by:  grabar_confFecha()      src/gregoriano.c

Global Function nuevo_nodo_xml()
Recibe un puntero a tipo xmlDocPtr, el nombre de la etiqueta del dato a añadir,
y el valor del dato. Crea un nuevo nodo xml en el archivo del primer parámetro
con la etiqueta del segundo y el valor del tercero.

void nuevo_nodo_xml ( xmlDocPtr doc, char* etiDato, char* valorDato )
Prototyped in:    src/xmlmisc.h
Calls:      xmlDocGetRootElement(), xmlNewChild()
Called by:  grabar_confFecha()      src/gregoriano.c
```

Cómo documentar código fuente en C con cxref.

```
Global Function obtener_dato_xml()
Recibe el nombre del documento, el nombre del nodo raíz y el nombre del nodo
cuyo valor se desea obtener. Devuelve un puntero xmlChar con el resultado. El
valor devuelto ha de ser liberado pasándolo como parámetro a la función xmlFree
().

xmlChar* obtener_dato_xml ( char* nomDoc, char* nomNodoRaiz, char* nomNodo )
Prototyped in:    src/xmlmisc.h
Calls:      mensaje()   src/gtkmisc.c
      strncat(),      strncpy(),      xmlDocGetRootElement(),      xmlFreeDoc(),
xmlNodeListGetString(), xmlParseFile(), xmlStrcmp()
Called by:  leer_confFecha()  src/gregoriano.c
      obtener_santoral()      src/gregoriano.c
```

Hay dos formas de utilizar cxref: incluirlo directamente en el Makefile o en la línea de comandos, con un script, por ejemplo.
La opción de incluirlo en el Makefile utilizando las autotools supondría una dependencia más a tener en cuenta en el configure.ac. Con un Makefile independiente, en cambio, debería funcionar sin mayores contratiempos.
Debido a esas razones me he decidido por el siguiente script:

```
cxref ./src/*.[ch] \
-Odoc -Nalmanaque -block-comments -xref-all -index-all -html \
-I. -I.. -I./src \
-I/usr/include/gtk-2.0 -I/usr/lib/gtk-2.0/include \
-I/usr/X11R6/include -I/usr/include/atk-1.0 -I/usr/include/pango-1.0 \
-I/usr/include/freetype2 -I/usr/include/glib-2.0 -I/usr/lib/glib-2.0/include \
-I/usr/include/cairo -I/usr/include/libxml2 \
```

La primera línea (aunque en realidad se trata de una sola, debido a los caracteres '\', pero para entendernos), es la llamada al programa diciéndole que procese todos los archivos que haya en el subdirectorio ./src que terminen en ".c" o en ".h".
La siguiente línea le dice que guarde la salida en el directorio llamado "doc" (-Odoc), que le dé al archivo principal el nombre de "almanaque" (-Nalmanaque), que no tenga en cuenta los caracteres '*' que pudiera haber al principio de cada línea dentro de un mismo comentario (-block-comments), que produzca todas las referencias cruzadas (-xref-all), que produzca un índice con todas las referencias cruzadas, (-index-all) y que la salida tenga el formato html. (-html).
A continuación hay que poner los directorios para los includes que necesitemos en los Makefiles para compilar, de la misma forma que se ponen para el gcc (tan simple como hacer copy-paste).

Debian produce un mensaje "warning" para cada archivo procesado diciendo que debe reconfigurarse como root :
Warning: cxref-cpp needs to be reconfigured against your latest gcc/cpp
Please run 'dpkg-reconfigure cxref' as root.

Ese mensaje no afecta para nada si todo ha sido bien hecho, aunque es cierto que hay que reconfigurar cxref tras cada actualización del compilador gcc, pero ya se encarga de avisar el propio cxref, produciendo unas salidas de errores bastante más abultadas si no nos hemos acordado.

Como más vale un ejemplo práctico que mil palabras de más, me voy a permitir remitir a quienes puedan estar interesados a la última versión de mi sempiterno almanaquillo (0.6.55) en la que he incluído el directorio "doc" en el paquete a distribuir, donde están los archivos html generados por cxref a partir del script de antes.

Enlaces:

http://www.gedanken.demon.co.uk/cxref/

http://www.fentlinux.com/listing/almanaque/almanaque-0.6.55.tar.gz

Por shicefgo, enero de 2006.