FlexCat

El generador flexible de catálogos

Versión 1.2

Jochen Wiedmann

Am Eisteich 9
72555 Metzingen (Deutschland)
Tel. 07123 / 14881
Internet: wiedmann@uni-tuebingen.de

La traducción a Castellano ha sido realizada por Antonio J. Gomez Gonzalez

1 Vistazo

A partir del Workbench 2.1 el Amiga ofrece un sistema bastante cómodo para usar programas en diferentes idiomas: La `locale.library`. (A esto se le llama localización, que es para lo que vale el nombre.)

La idea es sencilla: Eliges un idioma, el castellano la mayoría de los casos, y escribes tu programa de la misma forma que que lo hacías sin localización a excepción de que las cadenas constantes se substituyen por ciertas llamadas a función. Otra llamada a una función permite que los usuarios elijan otro idioma al iniciar el programa. (Esta última llamada lee un fichero externo, el llamado `catálogo`, y hace que se lean las cadenas o textos del catálogo en vez de las predefinidas).

Estos catálogos son independientes del programa. Todo lo que necesitas para añadir otro idioma es crear un nuevo catálogo, lo cual se puede hacer en cualquier momento sin falta de modificar el programa.

Sin embargo hay tareas adicionales para el programador: Necesita crear los catálogos, las cadenas predefinidas, y algo de código fuente para manejarlas. (Las funciones que se mencionaron antes). Flexcat está diseñado para hacer esto de una forma sencilla y casi automática sin perder flexibilidad, especialmente al crear el código fuente. Lo veremos más claro con un ejemplo:

Supongamos que queremos escribir un '`HolaMundoLocal.c`'. Nuestro programa final quedaría como:

```
#include <stdio.h>
#include <stdlib.h>
#include <HolaMundoLocal_Cat.h> /* ¡Debes incluirlo! */
#include <clib/exec_protos.h>

struct Library *LocaleBase;

void main(int argc, char *argv[])
{ /* Abre la librería tú mismo aunque el compilador permita la    */
  /*  apertura automática. NO salgas si falla OpenLibrary, en ese */
  /*  caso usaremos las cadenas internas.      */
  LocaleBase = OpenLibrary("locale.library", 38);

  OpenHolaMundoLocalCatalog(NULL, NULL);

  printf("%s\n", GetHolaMundoLocalString(msgHola));

  CloseHolaMundoLocalCatalog();
  if (LocaleBase)
     CloseLibrary(LocaleBase);
}
```

Fíjate que es casi igual que el 'HolaMundo.c' original a excepción de que sustituye la cadena "¡Hola mundo!" por una llamada a una función y que contiene algunas inicializaciones adicionales.

El programa anterior usa la constante msgHola. La llamada a GetHolaMundoLocalString hace la sustitución por la cadena correspondiente. Estas constantes y cadenas están definidas en el fichero Descripción de Catálogo. (see Chapter 4 [Descripcion de catalogo], page 5). Siempre se empieza creando un fichero de ese tipo, el 'HolaMundoLocal.cd', que podría ser como el siguiente:

```
; Por supuesto, se permiten comentarios. Toda línea que empieze
; con un punto y coma se supone un comentario.
;
; El idioma de las cadenas internas:
#language español
;
; Versión del catálogo, se usa en la llamada a Locale/OpenCatalog().
; Esto es diferente a Exec/OpenLibrary(): 0 significa cualquier
; versión de catálogo, ¡otro, significa coincidir exactamente!
#version 0
;
; Esto define una cadena y el ID que permite usarla.
; El número 4 indica que la cadena no debe tener menos de 4
; caracteres.
msgHola (/4/)
¡Hola mundo!
```

Usando FlexCat puedes crear otros dos ficheros a partir de la descripción de catálogo: el fichero include 'HolaMundoLocal_Cat.h' que define las constantes (ID), y el 'HolaMundoLocal_Cat.c', el cual contiene un vector con las cadenas y las funciones OpenHolaMundoLocalCatalog(), GetHolaMundoLocalString() y CloseHolaMundoLocalCatalog(). No necesitas saber como son, sólo cómo usarlas.¡ Especialmente si no necesitas saber nada sobre la locale.library!

En cambio, podrías estar interesado en el funcionamiento de estos ficheros o incluso podrías querer cambiarlos. Esta es la diferencia entre FlexCat y los demás generadores de catálogos: FlexCat no necesita usar un formato interno especial para la creación de esos ficheros. En su lugar usa ficheros externos, las Descripciones de fuente. Esto permite, por ejemplo, el uso de catálogos con AmigaDOS 2.0. see Chapter 6 [Descripcion fuente], page 8. Si usas las descripciones de código fuente de la distribución de FlexCat puedes crear los ficheros fuente con los siguientes comandos:

```
'FlexCat HolaMundoLocal.cd HolaMundoLocal_Cat.c=C_c_V21'
'FlexCat HolaMundoLocal.cd HolaMundoLocal_Cat.h=C_h.sd'
```

Una vez que tengas tu programa listo usarás FlexCat de nuevo para crear los ficheros Traducción de Catálogo, uno para cada idioma que quieras soportar. (Excepto para español, que es interno). See Chapter 5 [Traduccion de catalogo], page 7. Vamos a crear una traducción de catálogo en inglés.

```
'FlexCat HolaMundoLocal.cd NEWCTFILE English.ct'
```

Este fichero sería como sigue:

```
## version
## language
## codeset 0
; Por supuesto, se permiten comentarios. Toda línea que empieze
; con un punto y coma se supone un comentario.
;
; El idioma de las cadenas internas:
;
; Versión del catálogo, se usa en la llamada a Locale/OpenCatalog().
; Esto es diferente a Exec/OpenLibrary(): 0 significa cualquier
; versión de catálogo, ¡otro, significa coincidir exactamente!
;
; Esto define una cadena y el ID que permite usarla.
; El número 4 indica que la cadena no debe tener menos de 4
; caracteres.
msgHola

; ¡Hola mundo!
```

Como ves, se parece mucho a la descripción de catálogo. FlexCat incluye los comentarios de la descripción de catálogo, incluso los que no tienen mucho sentido: Fíjate en el comentario de la longitud de cadena, el cual no debería aparecer ahí ya que esa información sólo debe estar en la descripción de catálogo. Todo lo que tienes que hacer ahora es rellenar la información sobre la versión (se espera una cadena típica de versión como '`$VER: English.catalog 1.0 (22.05.94)`', el idioma de la traducción de catálogo (aquí para inglés sería '`english`'), el codeset (que debería ser siempre 0 por ahora, para más detalles mira en Locale/Catalogs()) y, por supuesto, las propias cadenas. FlexCat incluye las cadenas originales en forma de comentarios de forma que siempre sepas que es lo que tienes que poner.

Finalmente, creas los catálogos con comandos como:

```
'FlexCat HolaMundoLocal.cd English.ct CATALOG English.catalog'
```

Fíjate que ¡no necesitas el programa o los ficheros fuentes creados con FlexCat para los catálogos! Puedes crear nuevos catálogos en cualquier momento. Es usual ofrecer distribuciones con un fichero CatalogoNuevo.ct, de forma que los usuarios puedan crear sus propios catálogos.

Pero, ¿qué ocurre si cambias el programa más tarde? Simplemente edita la descripción de catálogo y usa FlexCat para actualizar las traducciones de catálogo:

```
'FlexCat HolaMundoLocal.cd English.ct NEWCTFILE English.ct'
```

Todo lo que tienes que hacer ahora es introducir las nuevas cadenas si es necesario.

2 Instalación

FlexCat está escrito en Ansi-C puro (excepto la localización), por ello, debería correr en cualquier Amiga y con suerte en otras máquinas después de compilarlo. (La localización queda como comentarios en ese caso). Esto tambien es aplicable a los programas: Flexcat está escrito utilizándose a sí mismo. Todas las descripciones de fuente distribuidas deberían crear programas que se ejecuten en cualquier Amiga, e incluso en cualquier máquina. (Por supuesto, debes asegurarte de que la variable LocaleBase tiene un valor '`NULL`' en este último caso). Sin embargo, la localización sólo es posible a partir del Workbench 2.1 porque la `locale.library` no estaba disponible antes.

No es imposible ofrecer localización sin la `locale.library`: Los ficheros de descripción de fuente '`C_c_V20.sd`' y '`C_h_V20.sd`' ofrecen un ejemplo en el que se usa la `iffparse.library` para sustituir la `locale.library` si ésta no está disponible. Esto permite Localización en el Workbench 2.0. See Section 7.1 [C], page 11.

Instalar FlexCat es simple: Copia el programa en un directorio en tu camino de búsqueda y elige un lugar para las descripciones de fuente que necesites. (Éstas son los ficheros que tienen nombres de la forma '`xx_yy.sd`', donde '`xx`' es el lenguaje de programación). Si quieres usar FlexCat en otro idioma distinto del inglés tambien necesitas copiar los catálogos correspondientes. Ej. para el castellano debes copiar '`Catalogs/español/FlexCat.catalog`' en '`Locale:Catalogs/español/FlexCat.catalog`' o '`PROGDIR:Catalogs/español/FlexCat.catalog`', donde '`PROGDIR:`' es el directorio del programa FlexCat. See Chapter 7 [Usando fuentes FlexCat], page 10.

3 Llamando a FlexCat desde el CLI

Flexcat es un programa basado en el CLI y no funciona desde el Workbench. Su sintaxis de llamada es

```
FlexCat CDFILE/a,CTFILE,CATALOG/k,NEWCTFILE/k,SOURCES/m
```

donde el significado de los argumentos es

CDFILE es el nombre de la descripción de catálogo a leer. Siempre es necesario. Señalar que el nombre base de la descripción de fuente se crea de éste distinguiendo entre mayúsculas y minúsculas. See Chapter 6 [Descripcion fuente], page 8.

CTFILE es el nombre del fichero traducción de catálogo que se leerá. Se necesita para la creación de catálogos o la actualización de una traducción de catálogo antigua usando el argumento NEWCTFILE: FlexCat lee el fichero viejo y la descripción de catálogo, y crea un fichero traducción de catálogo nuevo conteniendo las cadenas viejas y, posiblemente, líneas vacías para las cadenas nuevas.

CATALOG
es el nombre del fichero catálogo que se creará. Este argumento necesita que también se indique el argumento CDFILE.

NEWCTFILE
: es el nombre del fichero traducción de catálogo que se creará. FlexCat lee, si se dá, cadenas de CTFILE, y las cadenas que falten de la traducción de catálogo se sustituyen con líneas vacías. (La nueva traducción de catálogo sólo contendrá líneas vacías como cadenas si se omite el CTFILE).

SOURCES
: son los nombres de los ficheros de código fuente que se van a crear. Se debería poner en forma de '`fuente=patrón`' donde '`fuente`' es el fichero a crear y '`patrón`' es el nombre del fichero de descripción de fuente que se analizará.

Ver Chapter 1 [Vistazo], page 1 para ejemplos de líneas de comandos.

4 Ficheros de descripción de catálogo

Un fichero descripción de catálogo contiene cuatro tipos de líneas.

Líneas de comentario
: Cualquier linea que empieze por un punto y coma se supone una línea de comentario, y por tanto se ignora. (Las siguientes líneas de cadena son una excepción y pueden empezar con un punto y coma).

Líneas de comando
: Cualquier línea que empieze con un '#' (con la misma excepción que antes) se suponen líneas de comando. Los posible comandos son:

`#language <cad>`
: indica el idioma por defecto del programa, el idioma de las cadenas de la descripción de catálogo. Por defecto es '`#laguage english`'.

`#version <num>`
: indica el número de versión de los catálogos a abrir. Señalar que este número debe coincidir exactamente y no ser el mismo o superior como en *Exec/Openlibrary*. Una excepción es el número 0, que acepta cualquier catálogo. El valor por omisión es '`#versión 0`'. En `Locale/OpenCatalog` encontrarás más información sobre el idioma del catálogo y la versión.

`#lengthbytes <num>`
: Indica a Flexcat que ponga el número de bytes dado antes de cada cadena que contenga su longitud. La longitud es el número de bytes de la cadena sin bytes de longitud ni el byte '`NUL`' del final. (Los ficheros catálogo, y por tanto las cadenas del catálogo siempre tendrán un byte '`NUL`' al final. Esto no siempre es cierto para las cadenas por defecto, depende del fichero de descripción de fuente). '`<num>`' debe estar entre 0 y sizeof(long)=4, por omisión es '`#lengthbytes 0`'.

`#basename <cad>`

Pone el nombre-base de la descripción de fuente. See Chapter 6 [Descripcion fuente], page 8. Esto anula el nombre-base del argumento CDFILE de la línea de comandos. See Chapter 3 [Inicio del programa], page 4.

En los comandos no se distinguen mayúsculas de minúscalas.

Líneas de descripción

declaran una cadena. Son de la forma '`IDCAD (id/longmin/longmax)`' donde '`IDCAD`' es un identificador (cadena que consta de los caracteres a-z,A-Z y 0-9), '`id`' es un número único (desde ahora lo llamaremos ID), '`longmin`' y '`longmax`' son la longitud mínima y máxima respectivamente de la cadena. Los tres últimos se pueden omitir (¡aunque no los caracteres '`(//)`'!), en cuyo caso FlexCat elige un número y no restringe la longitud de la cadena. Lo mejor es no usar los IDs si no los necesitas. Las líneas que las siguen son

Líneas de cadena

contienen la propia cadena y nada más. Pueden contener ciertos caracteres de control que empiezan con una barra inversa:

'`\b`' Borra atrás (Ascii 8)

'`\c`' CSI (Introductor de Sequencia de Control) (Ascii 155)

'`\e`' Escape (Ascii 27)

'`\f`' Salto página (Ascii 12)

'`\g`' Pitido pantalla (Ascii 7)

'`\n`' Salto de línea, (newline) (Ascii 10)

'`\r`' Retorno de Carro (Ascii 13)

'`\t`' Tabulador (Ascii 9)

'`\v`' Tabulador Vertical (Ascii 11)

'`\)`' El paréntesis final que puede necesitarse como parte de una sequencia '`(..)`', ver Chapter 6 [Descripcion fuente], page 8.

'`\\`' La propia barra inversa.

'`\xHH`' El caracter dado por el código ASCII '`HH`', donde '`HH`' son dígitos hexadecimales.

'`\OOO`' El caracter dado por el código ASCII '`OOO`', donde '`OOO`' son dígitos octales.

Finalmente, una barra inversa sóla al final de la línea provoca la concatenación con la siguente línea. Esto permite usar cadenas de cualquier longitud, FlexCat no hace suposiciones sobre la longitud de la cadena.

Por tanto, una cadena se dá con una línea de descripción seguida de un línea de cadena. Veamos un ejemplo:

```
msgHola (/4/)
¡Hola, esto es castellano!\n
```

Aquí se omite el ID, por lo que FlexCat elige un número apropiado. El número 4 indica a FlexCat que la cadena siguiente no debe tener menos de 4 caracteres, y que puede ser de cualquier longitud. Mira en el fichero '`FlexCat.cd`' para más ejemplos.

5 Ficheros traducción de catálogo

Los ficheros traducción de catálogo son bastante parecidos a las descripciones de catálogo, a excepción de comandos diferentes y por no tener información sobre el ID de cadena ni sobre la longitud. (Éstos se toman de la descripción de catálogo). Deben aparecer todas las cadenas de la descripción de catálogo, (sin embargo, FlexCat no escribe en el catálogo las cadenas que son idénticas a la cadena por omisión), y no debe tener identificadores adicionales. Esto se puede asegurar fácilmente si se usa FlexCat para crear las traducciones de catálogo nuevas. See Chapter 1 [Vistazo], page 1.

Los comandos que se permiten en traducciones de catálogo son:

`##version <cad>`
: Indica la versión del catálogo en forma de cadena de Versión de AmigaDOS. Ejemplo:

  ```
  '##version $VER: English.ct 8.1 (22.05.94)'
  ```

  El número de versión de este catálogo es 8. De hecho, la versión de la descripción de catálogo debe ser 0 u 8.

`##language <cad>`
: El idioma del catálogo. Por supuesto, debe ser otro idioma distinto del idioma de la descripción de catálogo. Los comandos '`##language`' y '`##version`' deben estar presentes en la traducción de catálogo.

`##codeset <num>`
: Actualmente no se usa, debe ser 0. Este es el valor por defecto.

La cadena de antes sería algo como lo siguiente en la traducción de catálogo:

```
msgHola
Hello, this is english!\n
```

Mira en '`Español.ct`' para más ejemplos de una traducción de catálogo.

6 Ficheros de descripción de fuente

Esta es la parte especial de FlexCat. Hasta ahora no hay nada que no puedan ofrecer CatComp, KitCat u otros. El código fuente creado debe hacer fácil el uso de los catálogos sin perder flexibilidad. Debería poder utilizarse cualquier lenguaje de programación y debería poder satisfacerse cualquier requisito. Esto parece una contradición, pero la solución de FlexCat son los ficheros descripción de fuente que contienen un patrón del código fuente que se creará. Éstos son editables de la misma forma que lo son los ficheros descripción de catálogo y traducción de catálogo, por ello, FlexCat puede crear cualquier código.

Se analizan las descripciones de fuente para encontrar ciertos símbolos que se substituyen por ciertos valores. Símbolos posibles son los caracteres de barra inversa anteriores y, además, secuencias que empiezen con '`%`'. (Esto lo conocen bien los programadores en C).

'`%b`' es el nombre base de la descripción de catálogo. See Chapter 3 [Inicio del programa], page 4.

'`%v`' es el número de versión de la descripción de catálogo. No lo confundas con la cadena de versión de la traducción de catálogo.

'`%l`' es el idioma de la descripción de catálogo. Señalar que ésta se inserta como una cadena. Mira '`%`' más adelante.

'`%n`' es el número de cadenas de la descripción de catálogo.

'`%%`' es el propio caracter '`%`'.

Pero lo más importante son las siguientes secuencias. Éstas representan a las cadenas del catálogo de diferentes formas. Las líneas que contienen uno o más de estos símbolos se repiten para cada cadena.

'`%i`' es el identificador de la descripción de catálogo.

'`%d`' es el ID de la cadena.

'`%s`' es la propia cadena; se insertará dependiendo del lenguaje de programación, y se puede controlar con los comandos '`##stringtype`' y '`##shortstrings`'.

'`%(...)`' inserta el texto entre los paréntesis en todas las cadenas menos en la última. Esto se necesitará probablemente en vectores si las entradas del vector se deben separar con comas pero la última no se debe seguir con una coma. Señalar que entre los paréntesis no se substituirán las secuencias '`%`'. Se permiten, sin embargo, las secuencias de barra inversa.

Las secuencias de control '`%l`' y '`%s`' crean cadenas. Aunque la forma en que queden las cadenas depende del lenguaje de programación. Ese es el motivo de que la descripción de fuente permita

líneas similares a las de la traducción de catálogo. Éstas deben empezar con el primer caracter de la línea y cada comando debe tener su propia línea. Los posibles comandos son:

`##shortstrings`
: hace que las cadenas más largas se dividan en varias líneas. Esto no siempre será posible o no estará implementado en FlexCat, y por ello, la opción por defecto es crear sólo una cadena, probablemente, bastante larga.

`##stringtype <tipo>`
: Indica a FlexCat cómo deben aparecer las cadenas. Los tipos posibles son:

  None No se crean caracteres adicionales. Se inserta una imagen de la cadena y nada más. No se pueden poner caracteres binarios (las secuencias con barra inversa).

  C crea cadenas de acuerdo con el C. Las cadenas se preceden y finalizan con el caracter '`"`'. Las cadenas se dividen usando secuencias '`"\`' al final de la línea y '`"`' al principio de la nueva línea. (La barra inversa se necesita en macros). Los caracteres binarios se insertan usando '`\OOO`'. See Section 7.1 [C], page 11.

  Oberon es como el tipo de cadena C, excepto por la barra final al final de la línea.

  Assembler Las cadenas se crean usando '`dc.b`'. Los caracteres ASCII legibles se preceden y siguen con el caracter '`'`', los caracteres binarios se insertan como '`$XX`'. See Section 7.3 [Ensamblador], page 12.

  E Las cadenas se preceden y siguen con el caracter '`'`'. Un '`+`' concatena cadenas que se reparten por varias líneas. Los caracteres binarios se insertan de la misma forma que en C.

Veamos un fragmento del fichero '`C_h.sd`' creando un fichero include para el lenguaje de programación C.

```
##stringtype C
##shortstrings

#ifndef %b_CAT_H    /* Nos aseguramos de que sólo se lea una vez. */
#define %b_CAT_H

/*  Leemos los demás includes */
#include <exec/types.h>
#include <libraries/locale.h>

/*  Prototipos */
extern void Open%bCatalog(struct Locale *, STRPTR);
extern void Close%bCatalog(void);
extern STRPTR Get%bString(LONG);
```

```
/*  Definiciones de los identificadores y sus IDs. */
/*  Esta línea se repetirá para cada cadena.     */
#define %i %d

#endif
```

7 Incluyendo fuentes de FlexCat en programas propios

Por supuesto, esto depende del tipo de código fuente que se desee crear, y por tanto de la descripción de fuente. De lo que estamos hablando aquí es de los ficheros descripción de fuente que se distribuyen con FlexCat. See Chapter 6 [Descripcion fuente], page 8.

Todas las descripciones de fuente deberían permitir el uso del programa sin la `locale.library`. Sin embargo, debe estar presente una variable llamada '`LocaleBase`' ('`_LocaleBase`' para ensamblador) e inicializarse con NULL o con una llamada a *Exec/OpenLibrary*. En el primer caso no es posible la localización a no ser que se use el fichero de descripción de fuente '`C_c_V20.sd`'. Éste permite la localización bajo 2.0 sustituyendo la `locale.library` por la `iffparse.library`. (Para ello debe estar presente e inicializada una variable '`IFFParseBase`' como con '`LocaleBase`'). See Section 7.1 [C], page 11. El programador no necesita conocer estas librerías a no ser que quiera crear sus propias descripciones de fuente.

Hay tres funciones, y llamarlas es bastante sencillo.

OpenCatalog (*locale, idioma*)

Esta función probablemente abrirá el catálogo. El argumento `locale` es un puntero a la estructura Locale y `idioma` es una cadena que contiene el nombre del idioma que se debería abrir. En la mayoría de los casos deberían ser ambos '`NULL`' o '`NIL`', respectivamente, ya que en otro caso se anulan los valores que predefine el usuario. Para más detalles mira en *Locale/OpenCatalog*.

Si el usario tiene '`español`' y '`Deutsch`' como idiomas por omisión y el nombre base del programa es '`XXX`', buscará los siguientes ficheros:

```
'PROGDIR:Catalogs/español/XXX.catalog'
'LOCALE:Catalogs/español/XXX.catalog'
'PROGDIR:Catalogs/Deutsch/XXX.catalog'
'LOCALE:Catalogs/Deutsch/XXX.catalog'
```

donde '`PROGDIR:`' es el directorio actual del programa. (Se puede cambiar el orden de '`PROGDIR:`' y '`LOCALE:`' para evitar los requeters del tipo '`Inserta volumen YYY`'.

OpenCatalog es de tipo void (para programadores en Pascal un procedimiento) y por tanto no devuelve nada.

GetString (*ID*)
: Devuelve un puntero a la cadena con ese ID de la descripción de catálogo. Por supuesto estas cadenas son propiedad de `locale.library` y no se deben modificadar.

  Podría ser útil un ejemplo. Cojamos la cadena de la descripción de catálogo del ejemplo, que se llamaba `msgHola`. Las descripciones de fuente declaran una cosntante '`msgHola`' representando el ID. Se podría imprimir en C usando:

```
printf("%s\n", GetString(msgHola));
```

CloseCatalog (*void*)
: Esta función libera el catálogo (que está reservado en RAM) antes de terminar el programa. Puedes llamar a esta fucnión en cualquier momento, incluso antes de llamar a OpenCatalog.

7.1 Fuentes de FlexCat en programas en C

El fuente en C cosiste en dos partes: Un fichero '`.c`' que debería compilar y linkar sin problemas, y un fichero include que debería incluirse desde cualquier parte del fuente que use cadenas de catálogo y el cual define los IDs como macros.

Hay dos versiones diferentes para la parte '`.c`': '`C_c_V21.sd`' es un versión bastante simple usando las funciones correspondientes de la `locale.library` y que permite la localización a partir del Workbench 2.1. Por otro lado la '`C_c_V20.sd`' substituye la `locale.library` con la `iffparse.library` si la primera no está disponible y lo está la útlima. Esto permite la localización para el Workbench 2.0 también. Los programas que usen ésta deberían tener una opción `Idioma` y dar el argumento correspondiente a '`OpenCatalog`'. Esta opción no se debería usar en 2.1 y posteriores, y por ello el argumento de idioma de '`OpenCatalog`' debería seguir siendo '`NULL`'.

Por supuesto, sería posible escribir una tercera versión usando catálogos con Ansi-C, pero no quiero soportar la 1.3 más.

Para diferenciar las funciones '`OpenCatalog`' y '`CloseCatalog`' de las funciones respectivas de `Locale` con los mismos nombres, y para permitir diferentes catálogos en un mismo programa, las funciones de FlexCat obtienen nombres ligeramente modificados: '`OpenXXXCatalog`' y '`CloseXXXCatalog`', donde '`XXX`' es el nombre base de la descripción de fuente. El concepto ha sido copiado de GadToolsBox y, según creo, parece bueno. See Chapter 6 [Descripcion fuente], page 8.

Los prototipos de las funciones son:

```
void OpenXXXCatalog(struct Locale *loc, char *idioma);
STRPTR GetXXXString(ULONG);
void CloseXXXCatalog(void);
```

Mira en `HolaMundoLocal.c` para ver un ejemplo. (see Chapter 1 [Vistazo], page 1)

7.2 Fuentes de FlexCat en programas en Oberon

Hay dos descripciones de fuentes diferentes: '`Oberon_V38.sd`' crea el fuente usando '`Locale.mod`' de Harmut Goebel. '`Oberon_V39.sd`' crea el fuente usando el '`Locale.mod`' distribuido con `AmigaOberon`.

Los prototipos de las funciones son:

```
XXX.OpenCatalog(loc: Locale.LocalePtr; idioma : ARRAY OF CHAR);
XXX.GetString(num: LONGINT): Exec.StrPtr;
XXX.CloseCatalog();
```

donde '`XXX`' es el nombre base de la descripción de fuente. See Chapter 6 [Descripcion fuente], page 8.

Finalmente veamos un ejemplo de fuente de FlexCat:

```
MODULE HolaMundoLocal;

IMPORT  x:=HolaMundoLocal_Cat; Dos;

BEGIN
  x.OpenCatalog(NIL, "");

  Dos.PrintF("%s\n", x.GetString(x.msgHola));
  (* El catálogo se cerrará automáticamente  *)
  (* al finalizar el programa.               *)
END CualquierCosa;
```

7.3 Fuente de FlexCat en programas en ensamblador

El fuente en ensamblador se crea para usarlo con el ensamblador de Aztec. No debría ser muy diferente a otros ensambladores y deberías ser capaz de implementar descripciones de fuente propias. El fuente consiste de dos partes: Un fichero '`.asm`' que debería ensamblarse y linkarse sin problemas, y un fichero include '`.i`' que define los IDs de las cadenas y debe incluirse en el porgrama que las use.

Como siempre, el resultado de la función se da en d0, y las funciones guardan los registros d2-d7 y a2-a7. OpenCatalog espera sus argumentos en a0 (un puntero a una estructura Locale) y a1 (puntero a la cadena del idioma), que deberían ser NULL en la mayoría de los casos. GetString espera el ID de cadena en d0.

Finalmente, veamos un programa de ejemplo usando fuentes de FlexCat:

```
*   HolaMundoLocal.asm
    include "XXX.i" ; Es obligatorio abrirlo. Contiene
    ; "xref OpenHolaMundoLocalCatalog", ...
    xref    _LVOOpenLibrary
    xref    _LVOCloseLibrary
    xref    _AbsExecBase

    dseg
LocNam: dc.b "locale.library",0
    dc.l    _LocaleBase,4 ; Debe estar con este nombre

    cseg
main:   move.l  #38,d0  ; Abre la locale.library
    lea LocName,a1
    move.l  _AbsExecBase.a6
    jsr _LVOOpenLibrary(a6)
*   NO salir si falla OpenLibrary
    sub.l   a0,a0 ; Abre el catálogo
    sub.l   a1,a1
    jsr OpenHolaMundoLocalCatalog

    move.l  #msgHola,d0  ; Obtiene puntero a la cadena
    jsr GetHolaMundoLocalString
    jsr PrintD0  ; y la imprime

Final:
    jsr CloseHolaMundoLocalCatalog ; Cierra el Catálogo
    move.l  _LocaleBase,a1    ; Cierra la locale.library
    move.l  a1,d0    ; este test es necesario para 1.3
    beq Final1
    jsr CloseLibrary
Final1:
    rts
    end
```

7.4 Fuentes de FlexCat en programas en E

E se diferencia drásticamente de otros lenguajes de programación en un punto: No puedes compilar módulos separados y luego linkarlos juntos. Todo el código fuente debe estar en un fichero. La mejor solución a este problema es usar EPP de Barry Wills. (Origen: Aminet, directorio '`dev/e`', disco de Fred Fish). Esto te permite integrar los fuentes creados con la descripción de fuente `E21b.sd` en una línea:

```
PMODULE 'xxx_cat'
```

donde xxx es el nombre-base de tu aplicación. Sin EPP necesitas insertar el fuente de FlexCat manualmente en tu propio fuente. Debes insertar el fuente después de tus propias definiciones,

y antes del primer procedimiento. (De otra forma estarías obligado a crear e insertar más de un fichero con código fuente de FlexCat).

Las funciones '`get_xxx_string`', '`open_xxx_catalog`' y '`close_xxx_catalog`' están en el código fuente creado. (Estos nombres ligeramente modificados se necesitan para permitir catálogos diferentes en un mismo programa). Señalar que (al contrario que en C, por ejemplo) ¡no debes llamar a get_xx_string antes de open_xx_catalog!

Un '`HolaMundoLocal.e`' usando EPP podría parecerse a:

```
    /* HolaMundoLocal.e  */

    PMODULE holamundolocal_cat

    PROC main()
    /* Abre Locale.library; ¡No salir, si falla!  */
    localebase := OpenLibrary('locale.library', 0)

    /* Abre el fichero catálogo.      */
open_holamundolocal_catalog(NIL, NIL)

WriteF('\s\n', get_holamundolocal_string(MSG_HOLA_MUNDO))

close_holamundolocal_catalog()
    ENDPROC
```

Próximo desarrollo de FlexCat

No espero mucho más desarrollo de FlexCat porque que pienso que ya es bastante completo. Por supuesto, estoy abierto a sugerencias, trucos o críticas. Especialmente, me ofrezco a incluir nuevos tipos de cadenas, ya que ésto se puede hacer con cambios mínimos.

Estaría muy agradecido si me enviarais cualquie nueva descripcione de fuente para poder incluirlas en próximas distribiciones. Cualquier lenguaje de programación, cualquier extensión, siempre y cuando se haya comprobado bien el código fuente en un programa real. Tambien apreciaría recibir nuevos catálogos. Es suficiente insertar las cadenas en el fichero '`NewCatalogs.ct`' que es parte de la distribución.

Créditos

Agradezco especialmente a:

Albert Weinert
: por KitCat, el predecesor de FlexCat que me hizo grandes cosas, pero que finalmente no era lo suficientemente flexible.

Reinhard Spisser und Sebastiano Vigna
: por la versión de texinfo para Amiga. Esta documentación está escrita utilizándolo. (La traducción también :-)).

The Free Software Foundation
: por la versión original de texinfo y muchas otras excelentes cosas.

Matt Dillon
: por DICE y especialmente por DME.

Alessandro Galassi
: por el catálogo italiano.

Lionel Vintenat
: por la descripción de fuente de E y su documentación, los catálogos en francés y por informar sobre errores.

The people of #AmigaGer
: por contestarme muchas preguntas estúpidas, y por la diversión, por ejemplo stefanb (Stefan Becker), PowerStat (Kai Hoffmann), \ ill (Markus Illenseer), Quarvon (Jürgen Lang), ZZA (Bernhard Möllemann), Tron (Mathias Scheler), mungo (Ignatios Souvlatzis), \ jow (Jürgen Weinelt) und Stargazer (Petra Zeidler).

Commodore
: por el Amiga y el Kickstart 2.0. Seguid desarrollando sobre él y siguiré siendo un usuario de Amiga durante los próximos 8 años. ;-)

La traducción a castellano de este manual, así como de los catálogos del programa han sido realizados por:


```
Antonio Joaquín Gomez Gonzalez
C/ Venezuela, 14 - 2 I
33213 Gijon - Asturias (ESPAÑA)
E-mail: u0868551@oboe.etsiig.uniovi.es (mínimo hasta Sept. 94)
```


Índice

C

D

E

F

I

O

R

T

U

V

W

Table of Contents