KitCat

COLLABORATORS			
	TITLE : KitCat		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		January 17, 2023	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

KitCat

1.1 KitCat.guide

```
                KitCat Version 37.43 - Dokumentation
************************************

 Das Programm KitCat ist dafür geeignet, eigene Programm um die Funktion
der Lokalisierung zu erweitern.

 Die Beschreibung bezieht sich auf Release 1.1b, Version 37.43 von KitCat.
Die Anleitung wurde am 26 September 1993 zum letzten Mal bearbeitet.

 Falls in dieser Dokumenation an manchen stellen ein doppelter Schrägstrich
rückwärts (Backslash) auftaucht, dann ist dies NICHT beabsichtigt aber
leider nicht zu vermeiden, dies ist den verschiedenen Versionen der
amigaguide.library zuzuschreiben.  Wenn in diesen Fällen eigentlich nur
ein Schrägstrich rückwärts angebracht wäre, dann soll dies auch so sein.


Unbedingt vor dem ersten Benutzen zu lesen.


                 Wichtig!

                 Quelltext
                Wichtig für das Verständis


                 Wofür?
                Dateiformate:


                 Katalogbeschreibung

                 Katalogübersetzung
                Was sonst noch Wissenswert ist:


                 Steuerzeichen

                 Formatzeichen
```

1.2 KitCat.guide/Wichtig!

Copyright und andere Sachen
****

Dieses Paket darf nach Absprache mit dem Autor auch in eine Sammlung von Entwicklerwerkzeugen aufgenommen werden.

Das Programm und die mitgelieferten Quelltexte dürfen nicht kommerziell verwertet werden.

1.3 KitCat.guide/Quelltext

Der mitgelieferte Quelltext
****

Ab der Version 1.1b von KitCat wird der komplette Oberon Quelltext des Programms mitgeliefert. Dieser ist mehr schlecht als recht dokumentiert aber es sollte reichen.

Wenn sich jemand sich berufen fühlt das Programm zu verbessern, dann soll er dies ruhig machen. Allerdings sollte er/sie bevor er eine neue Version davon rausbringt diese mir zukommen lassen, damit ich die Versionen und deren Funktionen zusammenfügen kann und somit das Versionwirwarr vermeiden will. Siehe
Autor
.

1.4 KitCat.guide/Wofuer

Wofür wendet man KitCat an?
****

KitCat ist ein Programm, mit dem es möglich ist Sprachkataloge zu erzeugen. Diese können dann mit Hilfe der locale.library (1) in eigenen Programmen verwendet werden. So ist es möglich, auf relativ einfach Weise, seine Programme in mehreren Sprachen anzubieten.

Weitere Vorteile der Lokalisierung von Programmen:

* Es ist nicht nötig, die Zeichenketten in verschiedenen Sprachen mit in das eigentliche Programm aufzunehmen.

* Wenn ein Fehler im Programm behoben wird, dann ist er automatisch für alle anderen Sprachen behoben, d.h. man muss die Änderungen nicht in jeder Sprachversion nachvollziehen.

Ein Nachteil soll an dieser Stelle aber nicht verschwiegen werden:

* Ein etwas höherer Aufwand bei der Entwicklung des Programms.

Um ein Programm zu lokalisieren ist es nötig, sich eine Katalogbeschreibung zu erstellen. In der Katalogbeschreibung stehen die Zeichenketten in der Sprache, die in dem Programm eingebaut sein soll; mit

diesen Zeichenketten werden dann auch die Quelltexte erstellt. Die Katalogbeschreibung beinhaltet dann noch die Ausmaße, die eine Zeichenkette annehmen muss bzw. darf. Siehe
Katalogbeschreibung
.

Für einen Sprachkatalog braucht man eine Katalogübersetzung. Eine Vorlage der Katalogübersetzung kann von KitCat erstellt werden. In der Katalogübersetzung stehen die übersetzen Zeichenketten. Diese werden allerdings nicht von KitCat erzeugt. (Ein Übersetzungsprogramm einzubauen, wäre etwas zu viel erwartet.) Siehe
Katalogübersetzung
.

Außer für die eingebaute Sprache muß für jede weitere ein eigener Sprachkatalog erstellt werden. Da aus einer Katalogübersetzung nur ein Sprachkatalog erzeugt werden kann, muß für jeden Sprachkatalog eine Katalogübersetzung vorhanden sein.

---------- Fußnoten ----------

(1) Die locale.library ist ab AmigaOS 2.1 fester Bestandteil der Workbench

1.5 KitCat.guide/Katalogbeschreibung

Aufbau der Katalogbeschreibungs-Datei
****

Die Katalogbeschreibung ist eine Datei die nur aus ASCII-Zeichen bestehen darf. Diese sollten Sie mit einem Texteditor erstellen.

Eine Zeichenkette wird folgendermaßen definiert:

```
;
Bezeichner (Bezeichnerkennung/Minimale_Länge/Maximale_Länge)
Zeichenkette
;
```

* Bezeichner

  Ein nicht zu langer aber doch aussagekräftiger Name; mit Hilfe dieses Namens wird die Zeichenkette später aus dem Programm heraus adressiert.

  Der Name wird so in den Quelltext übernommen wie Sie ihn eingegeben haben, aus diesem Grunde sollten Sie darauf achten das der Name von der Programmiersprache in der sie das Programm schreiben auch übersetzt werden kann (1).

  Wenn Sie in C programmieren sollten Sie noch darauf achten, daß der Name eindeutig ist, d.h. der Name darf im gesamten Programm nur einmal definiert werden. Bei Oberon brauchen Sie nur darauf zu achten daß der Name im erzeugten Modul nur einmal definiert wird.

* Bezeichnerkennung

  Eine Integerzahl die die Nummer die Bezeichners angibt. Mit dieser Nummer werden die Zeichenketten im Quelltext und im Sprachkatalog abgelegt und auch wieder identifiziert. Der Bezeichner erhält den Wert der Bezeichnerkennung.

  Wenn man hier einen leeren Eintrag angibt, so wird als Bezeichnerkennung die Nummer vom vorherigen Bezeichner genommen und um eins erhöht, dieser Wert wird dann dem Bezeichner zugewiesen.

* Minimale_Länge

  Die Länge die eine Zeichenkette aufweisen muss.

  Kann leergelassen werden, dann wird als minimale Länge 0 angenommen.

* Maximale_Länge

  Die Länge die eine Zeichenkette nicht überschreiten darf.

  Kann leergelassen werden, dann darf eine Zeichenkette beliebig lang sein. Manche Compiler (2) können mit den erzeugten Quelltexten Probleme bekommen wenn die Zeichenkette zu lang wird. Es wird dann empfohlen, falls es notwendig sein sollte, die Zeichenketten in mehrere kleinere Zeichenketten aufzuteilen. Wenn Sie einen dieser Compiler besitzen und sie möchten vor der Compilierung erfahren ob die Zeichenkette zu lang für den Compiler ist, dann geben sie hier als maximale Länge 1024 an.

Es können beliebig viele Zeichenketten definiert werden.

Kommentarzeilen haben als erstes Zeichen ein Semikolon ;, diese werden von KitCat im Normalfall ignoriert, aber diese Kommentarzeilen werden nicht übernommen, wenn die Vorlage einer Katalogübersetzung erstellt wird.

Für die Katalogbeschreibung gibt es noch Steuerbefehle, die sich auf den erzeugten Quelltext auswirken. Die Steuerbefehle beginnen mit einem Doppelkreuz # gefolgt von dem Befehlsnamen (ohne Leerstelle). Nach dem Befehlsnamen folgen die Befehlsparameter, die keine Anführungszeichen " enthalten dürfen.

Folgende Steuerbefehle werden von KitCat unterstützt.:

#catalogname <katalogname>
  Der Name des Sprachkataloges der geöffnet werden soll, dieser wird im Quelltext eingetragen. Hier darf aber kein Suffix angehangen werden.

  Beispiel:

```
#catalogname FooBar
```

  Der Suffix .catalog wird von KitCat wo es nötig ist selbsttätig angehangen.

  Wenn dieser Eintrag nicht vorhanden ist, dann wird der Name der

Katalogbeschreibung dafür verwendet, wobei ein dort evtl. vorhandener Suffix entfernt wird.

#function <prozedurname>
Der Name der Prozedur welche die Zeichenketten anhand des Bezeichners heraussucht und an das Programm zurückgibt.

Die Standardeinstellung für die C-Quelltexte ist Get%sString, wobei das %s im Quelltext durch den Katalognamen ersetzt wird. Für den Oberon-Quelltext wird GetString benutzt.

Beispiel:

```
#function GetFoo
```

#basename <basisname>
Wenn man ein größeres Projekt entwickelt, dann ist es evtl. sinnvoll alle Sprachkataloge von seinen Programmen, die zu diesem Projekt, gehören in einem Unterverzeichnis zu legen so, das man als Anwender weiss wozu die Sprachkataloge dienen, als Beispiel ist hier das sys-Verzeichnis im "LOCALE:Catalogs/<sprache>" Verzeichnis zu nennen, hier liegen alle Sprachkataloge die zum System gehören.

Wenn man hier jetzt z.B. #basename oberon einträgt, dann werden die Sprachkataloge in den Verzeichnissen PROGDIR:Catalogs/<sprache>/oberon/ (PROGDIR: ist das aktuelle Verzeichnis des Programms) und LOCALE:Catalogs/<sprache>/oberon/ gesucht und von dort geöffnet.

Wobei <sprache> die Sprache darstellt in der der Sprachkatalog geöffnet werden soll.

#array <arrayname>
In C-Quelltexten kann es zu Bezeichnerüberschneidungen, mit eigenen und von KitCat generierten Programmmen kommen, deshalb ist es hier möglich einen eigenen Bezeichnernamen für das Array in dem die Zeichenketten abgelegt werden anzugeben.

Voreingstellt ist hier KC%sArray, wobei %s durch den Katalognamen ersetzt wird.

#arrayopts <arraytyp>
Es kann evtl. sinnvoll sein den Typ des Arrays zu ändern, voreingestellt ist hier static const, was aber nur für den C-Quelltext möglich ist.

#language <sprache>
Die Sprache in der die Zeichenketten aus Katalogbeschreibung erstellt wurde, dies ist zwingend notwendig. In den Quelltexten wird dies benutzt im festzustellen ob ein Sprachkatalog geladen werden muss oder nicht. Es ist wichtig das der Name in Kleinbuchstaben angegeben wird.

Beispiel:

```
#language english
```

#version <versionsnummer>
Die Version der Sprachdateien die für die korrekte Nutzung des

Programm vorhanden sein muss.

Jedem Sprachkatalog wird über die Katalogübersetzung eine Version gegeben, dies ist nötig, damit überprüft werden kann, ob der Sprachkatalog mit den Zeichenketten im Programm übereinstimmt.

Im Gegensatz zu einer Library-Version ist es hier nötig, die genaue Version anzugeben. Eine Ausnahme ist hier nur die Version 0: Wenn diese übergeben wird, dann wird jeder Version eines Sprachkatalogs geladen.

Beispiel:

```
#version 2
```

#lengthbytes <anzahl bytes>

Die Anzahl der Bytes in der die Länge vor der eigentlichen Zeichenkette im Quelltext und im Sprachkatalog stehen soll.

Beispiel:

```
#lengthbytes 2
```

Hier würde die Zeichenkette im Quelltext so definiert werden.

```
msgHello  = "\x00\x06Hello!";
```

Bei einer Länge von 0 werden keine Längenangabe vor die Zeichenkette angegeben.

Die Länge der Längenangabe gilt für die Zeichenketten die nach dem Befehl in der Katalogbeschreibung stehen.

Bei der Anforderung einer Zeichenkette wird nun nicht mehr der Zeiger auf die Zeichenketten zurückgegeben, sondern ein Zeiger auf die Länge und erst nach <anzahl bytes>-Bytes folgt die eigentliche Zeichenkette.

Eine Katalogbeschreibung könnte folgendermaßen aussehen.

```
; Katalogbeschreibung für das Programm : FooBar
#catalogname FooBar
#function GetString
#language deutsch
,
msgHello (/4/20)
Hallo!
;
msgGoodBye (//)
Ich Danke Ihnen für die Nutzung des Programms\n ich hoffe sie haben viel\
Spaß dabei gehabt.
;
menuLoad (100//)
Laden ...
;
menuSave (//)
Speichern ...
; Ende der Datei
```

```
 Bei der Erzeugung eines Quelltextes werden den Bezeichner folgende Werte
zugewiesen.

     msgHello   = 0;
     msgGoodBye = 1;
     menuLoad   = 100;
     menuSave   = 101;

 Eine Anwendung aus einem Programm, hier in Oberon, könnte folgendermaßen
aussehen:

       Dos.PrintF(FooBarLocale.GetString(FooBarLocale.msgGoodBye)^);

 Hier wird die Zeichenkette die dem Bezeichner msgGoodBye zugeordnet ist in
einem Shell-Fenster ausgeben.

 ---------- Fußnoten ----------

 (1)  In Oberon dürfen z.B. kein Unterstriche "_" verwendet werden

 (2)  Aztec C und ältrere SAS C Versionen erlauben nur begrenzt lange
Makros, und als solche werden die Zeichenkette in den Quelltexten definiert
```

1.6 KitCat.guide/Kataloguebersetzung

```
                Aufbau der Katalogübersetzungs-Datei
************************************

 Der Aufbau einer Katalogübersetzungs-Datei ist ähnlich wie der einer
Katalogbeschreibung. Allerdings werden dort nur der Bezeichner und die
Übersetzte Zeichenkette eingetragen. Wenn man mit KitCat aus dem zuvor
beschriebenen Beispiel (siehe
                Katalogbeschreibung
                ) eine Vorlage für eine
Katalogübersetzung erstellt, dann sieht diese so aus:

     ## version $VER: XX.catalog XX.XX (XX.XX.XX)
     ## codeset 0
     ## language xx
     ;;
     msgHello

     ;; Hallo!;
     ;;
     msgGoodBye

     ;; Ich Danke Ihnen für die Nutzung des Programms\n ich hoffe sie haben viel;
     ;; Spaß dabei gehabt.;
     ;;
     menuLoad

     ;; Laden ...;
     ;;
```

```
     menuSave

     ;; Speichern ...;
     ;;
```

In den ersten Zeilen stehen Befehle, deren Paramater mit in den Sprachkatalog übernommen werden. Diese Befehle beginnen mit einem ## und werden später genauer erklärt.

Die Übersetzte Zeichenkette muss direkt unter dem Bezeichner eingeben werden, hier gelten die gleichen Regeln wie bei der Katalogbeschreibung.

In den Kommentarzeilen hinter dem Bezeichner, die mit einem doppeltem Semikolon ;; beginnen, sind die Zeichenketten aus der Katalogbeschreibung. Sie werden bei jedem Erzeugen der Katalogübersetzung eingesetzt.

Beispiel einer übersetzten Katalogbeschreibung:

```
     ## version $VER: FooBar.catalog 1.0 (11.09.93)
     ## codeset 0
     ## language english
     ;;
     msgHello
     Hello!
     ;; Hallo!;
     ;;
     msgGoodBye
     Thank you für using this Prograann\n I hope you had much fun doing this.
     ;; Ich Danke Ihnen für die Nutzung des Programms\n ich hoffe sie haben viel;
     ;; Spaß dabei gehabt.;
     ;;
     menuLoad
     Load...
     ;; Laden ...;
     ;;
     menuSave
     Save...
     ;; Speichern ...;
     ;;
```

Folgende Kommandow werden zur Zeit von KitCat hinter dem ## Zeichen verstanden (andere werden ignoriert):

version <version>
: Standard Versionszeichenkette. Dieser Eintrag wird in den Sprachkatalog übernommen, wichtig ist hier, daß die Version mit der in der Katalogbeschreibung übereinstimmt, da sonst der Sprachkatalog nicht geöffnet werden kann.

  Beispiel:

  ```
  ## version FooBar.catalog 1.0 (13.09.93)
  ```

codeset <set>
: Zur Zeit noch keine Verwendung innerhalb der locale.library, allerdings sollte sie aus Kompatibilitätsgründen derzeit auf 0 gesetzt werden.

```
     Beispiel:

          ## codeset 0

language <sprache>
     Die Sprache in der die Katalogübersetzung erstellt worden ist, diese
     wird auch in den Sprachkatalog übernommen und beim Öffnen des
     Sprachkataloges überprüft.

     Beispiel:

          ## language english
```

1.7 KitCat.guide/Steuerzeichen

```
Steuerzeichen in Zeichenketten
******************************

 In einer Zeichenkette ist möglich, und auch oft nötig, bestimmte
Sonderzeichen einzugeben und darzustellen. Dies ist über sogennate
Steuerzeichen möglich.  Ein Steuerzeichen beginnt immer mit einem \  und
braucht nicht einzeln zu stehen. Folgende Steuerzeichen werden von KitCat
zugelassen.

     Zeichen  Bedeutung
       \a      display beep                (ASCII 7)
       \b      backspace                   (ASCII 8)
       \c,
       \[      control Sequence Introducer (ASCII 155)
       \e      escape                      (ASCII 27)
       \f      formfeed                    (ASCII 12)
       \n      newline                     (ASCII 10)
       \r      carriage return             (ASCII 13)
       \t      tab                         (ASCII 9)
       \v      vertical tab                (ASCII 11)
       \\      Das Zeichen \
       \xNN    Zeichen mit dem ASCII Code von NN in Hexadezimal
       \NNN    Zeichen mit dem ASCII Code von NNN in Octal

 Wenn eine Zeile mit einem \  endet, bedeutet dies daß die nächste Zeile
auch noch zu der aktuellen Zeichenkette gehört. Allerdings dürfen da keine
Kommentarzeilen drin vorkommen.
```

1.8 KitCat.guide/Formatzeichen

```
Formatzeichen in Zeichenketten
******************************

 An Formatzeichen werden die erlaubt, die von den Funktionen RawDoFmt() und
```

FormatString() erkannt und umgesetzt werden. Genaue Hinweise stehen in den Autodocs. Siehe RawDoFmt().

Damit die eigenen Programm auch noch unter OS 2.04 richtig laufenn, sollte man in der Katalogbeschreibung nur die Formatzeichen benutzen, die von Exec/RawDoFmt() erkannt werden. Dies gilt insbesondere bei der Angabe der Position innerhalb des Parameter-Arrays. In der Katalogübersetzung kann man ruhig Positionsparamter benutzen, denn der Katalog wird ja erst ab OS 2.1 geöffnet und dort wird Exec/RawDoFmt() durch eine Prozedur ersetzt die auch die Positionparameter versteht. Siehe FormatString().

1.9 KitCat.guide/Programmparameter

Programmparameter und Programmstart
****

KitCat hat folgende Befehlsschablone:

```
DESCRIPTOR,TRANSLATION,CTFILE/K,CATALOG/K,OFILE/K,CFILE/K,
L=LANGUAGE/K,V39/S,V38/S,SS=SHORTSTRINGS/S,NSS=NOSHORTSTRING
```

DESCRIPTOR
: Name der Katalogbeschreibungs-Datei, die verarbeitet werden soll. Muss unbedingt immer angegeben werden.

TRANSLATION
: Name der Katalogübersetzungs-Datei, die verabeitet werden soll.

CTFILE
: Katalogübersetzungs-Datei die erzeugt werden soll, wenn man beim erzeugen eine Übersetzungs-Datei eine Übersetzungdatei bei TRANSLATION angibt, so werden die Zeichenketten aus der schon vorhandenen Datei übernommen (wenn der Bezeichner stimmt). Es ist auch möglich die alte Datei im gleichen Aufruf wieder zu überschreiben.

CATALOG
: Der Sprachkatalog, der erzeugt werden soll.

OFILE
: Der Name des Oberon-Quelltextes der erzeugt werden soll. Wenn das .mod fehlen sollte, dann wird es automatisch drangehangen.

CFILE
: Der Name der C-Quelltexte der erzeugt werden soll. Auch hier ist es nicht nötig ein .c oder .h anzugeben, dies wird bei den erzeugten Quelltexten selbsttätig vorgenommen.

LANGUAGE
: Sprache, in der KitCat seine Textausgaben machen soll. Wenn nichts angegeben wird, dann wird die mit dem Landes-Voreinsteller eingestellte Sprache genommen. Die eingebaute Sprache ist deutsch. Wenn die Sprache nicht gefunden wird, dann wird auch die eingebaute Sprache genommen.

```
V39
     Hiermit kann man KitCat dazu bewegen, den Oberon-Quelltext so zu
     erzeugen, daß er sich auch mit dem V39 Locale.mod von Hartmut Goebel
     übersetzen läßt.

V38
     Gibt an, daß der Oberon-Quelltext für den V38 Locale.mod, welcher dem
     AmigaOberon Compiler 3.0 mitgeliefert wurde, zu erzeugen ist. Dies ist
     auch automatisch die Voreinstellung.

SHORTSTRINGS
     Wenn dieses gesetzt ist, dann wird bei der Quelltext Erstellung nicht
     die komplette Zeichenkette in einer Zeile definiert, sondern die
     Definition der Zeichenkette über mehrere Zeilen verteilt. Dies ist
     für die Leute, die es nicht lassen können, den Quelltext
     nachträglich zu ändern aber deren Editor nur 255 Zeichen pro Zeile
     zuläßt.

NOSHORTSTRINGS
     Hiermit wird angegeben, daß die Zeichenketten in den Quelltexten in
     einer einzigen Zeile erstellt werden. Dies ist voreingestellt.

 Da es allerdings bei manchen Parametern lästig ist, diese immer wieder
anzugeben, weil man persönlich z.B. immer nur V39 oder SHORTSTRINGS will,
kann man sich eine ENV-Variable anlegen, die diese Parameter enthält.

       SetEnv KitCat V39 SHORTSTRINGS

 Und wenn man dies dann auch noch dauerhaft speichern will, ist ein

       SetEnv ENVARC:KitCat V39 SHORTSTRINGS

 durchaus angebracht.
```

1.10 KitCat.guide/Vorgehensweise

```
                Vorgehensweise und Tips
**********************

 Um sein Programm zu lokalisieren, ist es notwendig, alle Zeichenketten mit
in die Katalogbeschreibung aufzunehmen, die in dem Programm benutzt werden.

 Als fiktives Beispiel nehmen wir jetzt das Programm FooBar welches die
eingebaute Sprache deutsch hat. Hier wird jetzt nicht nochmal der
Dateiaufbau erklärt, dieser wird in den Kapiteln Katalogbeschreibung und
Katalogübersetzung beschrieben. (Siehe
                Katalogbeschreibung
                , siehe

                Katalogübersetzung
                )

 Im weiteren Verlauf werden die verschiedenen Dateien mit verschiedenen
```

Suffixen angeben. Die Suffixe im einzelnen sind:

.cd
     Ist für eine Katalogbeschreibung (Englisch: Catalog Description).

.ct
     Ist für eine Katalogübersetzung (Englisch: Catalog Translation).

 Es ist besser, sich an die Suffixe zu halten, denn diese sind mal von Commodore so definiert worden und es wird damit für jeden, der sich ein bisschen damit beschäftigt hat klar, was dies für Dateien sind.

 Sie erstellen sich also eine Katalogbeschreibung mit dem Namen FooBar.cd, in diese schreiben sie alle Zeichenketten die in dem Programm vorkommen und geben ihnen eindeutige Bezeichnernamen. Wenn Sie die Zeichenketten eingeben haben, dann erstellen sie mit KitCat aus der Katalogbeschreibung den Quelltext für Ihre Programmiersprache, die Sie benutzen, für die genaue Anwendung der Quelltexte lesen Sie bitte die betreffenden Kapitel. (Siehe

                Oberon-Quelltext
                , siehe
                C-Quelltext
) Es ist jetzt noch nicht notwendig eine Katalogübersetzung zu erstellen, diese braucht man erst, wenn man Sprachkataloge erstellen möchte.

     KitCat FooBar.cd OFILE FooBarLocale.mod

 In diesem Beispiel wird ein Oberon-Quelltext erzeugt, als zu öffnender Sprachkatalog wird im Quelltext FooBar.catalog eingetragen. Der Name des Sprachkatalogs wird aus dem Namen der Katalogbeschreibung gebildet (indem ihm das vorhandene Suffix entfernt wird und .catalog angehangen wird), so daß sie hier auf die Groß- und Kleinschreibung des Names achten sollten.

 Jetzt binden sie den Quelltext in Ihr Programm ein. Gehen Sie hin und ersetzen Sie alle evtl. schon vorhandenen Zeichenketten durch den entsprechenden GetString() Aufruf. Denken Sie daran, daß sie die Zeichenketten nicht mehr verändern dürfen. Wenn Sie sie doch verändern müssen, dann müssen Sie sich die Zeichenketten vorher kopieren.

1.11 KitCat.guide/Oberon-Quelltext

Oberon-Quelltext
****

 Der Oberon-Quelltext ist so aufgebaut, daß das Programm ohne die locale.library arbeitet. Allerdings werden dann natuerlich nur die eingebauten Strings verwendet. Fuer den Programmierer sind keinerlei Kenntnisse der locale.library nötig. Es gibt lediglich 3 Funktionen, die eigentlich recht simpel sind. Ich gehe jetzt in der Erklärung der Funktionen davon aus, des es sich im das Programm foo handelt.

 Die Funktionen sind:

- : OpenCatalog (loc : Locale.LocalePtr; language : ARRAY OF CHAR)
Diese Funktion eröffnet den Katalog der zum Programm gehört. loc ist ein Zeiger auf eine Locale-Struktur und language ein String der die Sprache enthalten kann in der der Katalog geöffnet werden soll.

Diese werden an die Locale-Funktion OpenCatalog übergeben, näheres siehe OpenCatalog(). Gewöhnlich sollte der Aufruf einfach ein OpenCatalog(NIL, "") sein, hier wird dann der Katalog in der Sprach geöffnet wie er in dem Landes-Voreinsteller eingestellt ist.

Hat der Benutzer im Landes-Voreinsteller die Sprachen deutsch und francais eingestellt, so wird nacheinander nach folgenden Dateien gesucht:

```
PROGDIR:Catalogs/Deutsch/foo.catalog
LOCALE:Catalogs/Deutsch/foo.catalog
PROGDIR:Catalogs/Francais/foo.catalog
LOCALE:Catalogs/Francais/foo.catalog
```

Dabei ist PROGDIR: das Verzeichnis in dem sich das Programm befindet. Die Reihenfolge von PROGDIR: und LOCALE: kann vertauscht werden, falls letzteres gerade gemounted (wunderbares NeuDeutsch :-) ist und ersteres nicht.

OpenCatalog() liefert kein Ergebnis weil es für den Programmablauf nicht erforderlich ist, falls ein Sprachkatalog nicht geöffnet werden könnte so wird die eingebaute Sprache verwendet.

Falls sie in Ihrer Applikation während des Programm ablaufes die Sprach ändern wollen, so rufen sie einfach wieder OpenCatalog() auf, ein evtl. schon geöffneter Sprachkatalog wird dann von der Prozedur geschlossen.

- : GetString (num : LONGINT)
Ist der Katalog eröffnet, so liefert diese Funktion einen Zeiger auf den String mit der angegebenen Nummer. Für ID sollte dabei unbedingt das in der Katalogbeschreibung definierte Makro eingesetzt werden. Nehmen wir an, in der Katalogbeschreibung stünde der folgende Eintrag:

```
msgHello (/4/20)
Hallo!
```

Der dazugehörige String kann nun mit so ausgegeben werden:

```
Dos.PrintF( "%s\n", GetString( msgHello ) );
```

Beachten Sie, daß GetString() ein Zeiger auf einen String ist, dieser String darf auf keinen Fall geändert werden, da der Sprachkatalog nicht für jedes Programm geladen wird, sondern es kann wie bei einer Library von mehrere Processen gleichzeitig auf den Katalog zugegriffen werden. Ferner bleibt der Zeiger natürlich nur solange gültig, wie der Katalog eröffnet ist, das heißt bis zum Aufruf von CloseCatalog().

- : CloseCatalog ()
Mit dieser Funktion wird der Sprachkatalog wieder, so das er bei Bedarf von System aus dem Speicher entfernt werden kann.Es ist erlaubt, diese

```
     Funktion auch dann aufzurufen, wenn der zugehörige Aufruf von
     OpenCatalog() nicht erfolgreich war oder OpenCatalog() gar nicht
     aufgerufen wurde. Es ist in einer normalen Anwendung nicht nötig den
     Sprachkatalog selbst zu schliessen, dieser wird bei Beendigung des
     Programms automatisch geschlossen.

 Zum Schluß noch ein Beispiel eines Programms, welches den von KitCat
erzeugten Quelltext verwendet:

         MODULE KitCatTest;

         IMPORT
           l := KitCatTestLocale,
           Dos;

         BEGIN
           l.OpenCatalog( NIL, "" );
           Dos.PrintF( "%s\n", l.GetString( l.msgHello ) );
         END KitCatTest;
```

1.12 KitCat.guide/C-Quelltext

```
                C-Quelltext
***********

 Der C-Quelltext ist so aufgebaut, daß das Programm ohne die
locale.library arbeitet. Allerdings werden dann natuerlich nur die
eingebauten Strings verwendet. Fuer den Programmierer sind keinerlei
Kenntnisse der locale.library nötig, wichtig ist lediglich, daß diese vor
dem ersten Aufruf der KitCat-Funktionen eröffnet wird. Es gibt lediglich 3
Funktionen, die eigentlich recht simpel sind. Diese haben aber unter
Umständen unterschiedliche Namen: Es gibt nämlich für jeden zu
eröffnenden Katalog verschiedene Ausgaben derselben Funktionen. (Das Prinzip
ist von der GadToolsBox übernommen und meines Erachtens bewährt.) Im
Folgenden gehe ich davon aus, daß XXX der in der Katalogbeschreibung
eingestellte Basisname ist. Siehe
                Katalogbeschreibung
                .

 Die Funktionen sind:

 - : OpenXXXCatalog (struct  Locale  *loc,  STRPTR  language )
     Diese Funktion eröffnet den Katalog XXX.catalog. loc ist ein Zeiger
     auf eine Locale-Struktur und language ein Zeiger auf einen String.
     Diese werden an die Locale-Funktion OpenCatalog übergeben, näheres
     siehe dort. Gewöhnlich sollten beide Argumente NULL sein.

     Hat der Benutzer als Vorgabesprachen etwa "Deutsch" und "Francais"
     eingestellt, so wird nacheinander nach folgenden Dateien gesucht:
              PROGDIR:Catalogs/Deutsch/xxx.catalog
              LOCALE:Catalogs/Deutsch/xxx.catalog
              PROGDIR:Catalogs/Francais/xxx.catalog
              LOCALE:Catalogs/Francais/xxx.catalog
```

Dabei ist PROGDIR: das aktuelle Directory des Programms. Die Reihenfolge von PROGDIR: und LOCALE: kann vertauscht werden, falls letzteres gerade gemounted (wunderbares NeuDeutsch :-) ist und ersteres nicht.

OpenXXXCatalog ist vom Typ void, liefert also kein Ergebnis.

- : GetXXXString (LONG ID)
Ist der Katalog eröffnet, so liefert diese Funktion einen Zeiger auf den String mit der angegebenen Nummer. Für ID sollte dabei unbedingt das in der Katalogbeschreibung definierte Makro eingesetzt werden. Nehmen wir an, in der Katalogbeschreibung stünde der folgende Eintrag:

```
msgHello (/4/20)
Hallo!
```

Der dazugehörige String kann nun mit so ausgegeben werden:

```
printf("%s\n", GetXXXString (msgHello));
```

Damit das Makro definiert ist, muß natürlich die Datei XXX.h mit #include eingebunden werden.

Beachten Sie, daß GetXXXString vom Typ const char * ist! Der String darf nicht verändert werden! Ferner bleibt der Zeiger natürlich nur solange gültig, wie der Katalog eröffnet ist, das heißt bis zum Aufruf von CloseXXXCatalog().

- : CloseXXXCatalog (void)
Mit dieser Funktion wird der Katalog (das heißt das belegte RAM) wieder freigegeben. Es ist erlaubt, diese Funktion auch dann aufzurufen, wenn der zugehörige Aufruf von OpenXXXCatalog nicht erfolgreich war oder OpenXXXCatalog gar nicht aufgerufen wurde.

Zum Schluß noch ein Beispiel eines Programms, das KitCat verwendet:

```
#include <stdio.h>
#include <stdlib.h>
#include <XXX.h>
#include <clib/exec_protos.h>

struct Library LocaleBase;  /*  Library auf jeden Fall selber er-   */
                            /*  öffnen, auch wenn der Compiler das  */
                            /*  automatisch kann!                   */


void main(int argc, char *argv[])

{
  /*  KEIN Abbruch, falls OpenLibrary() schiefgeht! (Natürlich  */
  /*  nur, wenn keine Locale-Funktionen direkt benutzt werden.) */
  LocaleBase = OpenLibrary("locale.library", 38));
  OpenXXXCatalog(NULL, NULL);

  .... (andere Funktionen)

  printf("%s\n", GetXXXString(msgHello));

  .... (nochmal andere Funktionen)
```

```
          CloseXXXCatalog();
          if (LocaleBase != NULL)
          { CloseLibrary(LocaleBase);
          }
        }
```

1.13 KitCat.guide/Autor

```
Adresse des Autors
******************

 Falls Verbesserungsvorschläge zum Programm vorhanden sind oder Fehler
jedweder Art vorhanden sein sollten oder für neue Katalog-Übersetzungen
oder das Programm verbessert wurde bitte ich darum, mich zu kontaktieren.

 Dies ginge über die normale Post bzw. Telekom.

     Albert Weinert
     Krähenweg 21
     D-50829 Köl
     Tel: 0221 / 580 29 84
     Deutschland

 oder über die elektronische Post, die auf jeden Fall bevorzugt wird.

     Z-Netz: A.WEINERT@DARKNESS.ZER
     SubNet: aweinert@darkness.gun.de

 Der C-Quelltext, der von dem Programm erzeugt wird, ist von Jochen
Wiedmann.  Bei Fragen dazu wendet man sich an besten an ihn, denn dazu kann
ich wenig sagen.

 Auch hier geht dies über die normale Sackpost, oder 2-Draht-Verbindung.

     Jochen Wiedmann
     Am Eisteich 9
     72555 Metzingen
     Tel: 07123 / 14881

 Und im Zeitalter der Vernetzung hat auch er eine elekronische Postadresse
(die wohl auch bevorzugt wird). (Sehr richtig, JW! :-)

     Internet: wiedmann@mailserv.zdv.uni-tuebingen.de
```

1.14 KitCat.guide/Danksagungen

```
Warum ist das Programm enstanden und wie.
*****************************************
```

Da hat man das Problem, als Programmierer in einer noch sogenannten Außenseitersprache wie Oberon-2, daß diese von Commodore gar nicht unterstützt wird, so daß es für Oberonprogrammierer nicht die ganzen schönen Tools gibt, die es für C- und Assemblerprogrammier gibt. Dies ist irgendwie schade. Also bleibt einem nur der Weg des Wartens, bis sich irgendeiner aufrafft, etwas für die eigene Sprache macht oder es halt selber zu versuchen. Naja, ich habe mich dann nun dazu aufgerafft, es selber mal zu machen; KitCat kam dabei raus.

Nun ist es ja nicht so daß man in Bezug auf die Lokalisierung in Oberon nun komplett allein gelassen wurde, nein, da gab es im AmigaMagazin ein Programm womit man auch Oberonquelltext erzeugen konnte. Dieses Programm hieß MakeCat und war auch schön über ein GUI zu bedienen, aber dies hat sich auf Dauer, als erheblicher Nachteil erwiesen, jedenfalls für mich, da es eben nur über dieses GUI zu steuern war. Auf die restlichen Nachteile des Programms gehe ich jetzt hier nicht näher ein, schließlich ist das Programm auch frei verfügbar.

Anfänglich habe ich mir nur ein Programm geschreiben, das aus den Katalogbeschreibungen mir einen Oberon Quelltext erzeugt hat. Dies nannte sich Cd2Oberon, nur ich stellt mir die Frage, was machen die Programmierer die in Oberon programmieren, CatComp (das Pendant zu diesem Programm von Commodore) nicht haben und mit MakeCat auch nicht zurecht kommen? Also beschloß ich auch dafür etwas zu machen und erweiterte Cd2oberon um die Funktion des Erzeugens von Katalogübersetzungen . Und dann kam mal eine Frage im SubNet, wo nach einem Programm gefragt wurde, womit man seine Programme lokalisieren kann. Da erwähnte ich, daß ich so ein Programm entwickeln würde und ich noch Beta-Tester suche. Mit den Beta-Testern fanden sich dann auch eine ganze Reihe von Leuten, die auch einen C-Quelltext erzeugt haben wollten. Da sich unter den Betatestern auch einer fand, der den C-Quelltext erstellte, wurde dies in das Programm mit eingebaut. Die erzeugten Quelltexte entsprechen jetzt aber nicht den Erzeugnissen, die von MakeCat und CatComp erzeugt wurden. Nun ist das Programm eigentlich fertig, es hat nicht alle Funktionen von CatComp (Objektdateien erzeugen) und MakeCat (aus Sprachkatalogen können noch Katalogübersetzungen und Katalogbeschreibungen erstellt werden) aber es hat dafür noch die eine oder andere Funktion, die diese Programme nicht haben.

Danksagungen

Also mein Dank gilt erstmal den Betatesern

* Frank Dürring
  Er brachte so manchen Vorschlag mit in das Programm.

* Thomas Igracki
  Er kam spät dazu, und wollte unbedingt was reinhaben was ich nicht reinhaben wollte, dies ist zwar jetzt immer noch nicht drin, wird auch nie reinkommen, aber dafür ist etwas drin was ihm das ermöglicht, was er machen wollte. Aber sein Programm OOL hat sich im Laufe der Programmierung meiner Programme als absolut nützlich erwiesen, und ich will es heute nicht mehr missen.

* Holger Schemel

Der vor lauter Prüfung nicht dazu gekommen ist das Programm richtig zu testen.

* Jochen Wiedmann
  Er war wohl der aktivste Betatester. Von ihm stammt der erzeugte C-Quelltext, und die Dokumentation zum C-Quelltext, er brachte die meisten Vorschläge in das Programm mit ein (und hat noch ganz viele). Und das die Anleitung lesbar und verständlich ist das ist auch ihm zu verdanken, meine erste Niederschrift wurde von ihm redigiert.

Des weiteren will ich folgende Leute und Gruppen noch erwähnen.

* Thomas Esser
  Mein (nicht direkt meiner, sonder der der Serverbox wo ich als Point tätig bin) Sysop, der das ganze mit der Update-Verschickerei so einfach mitgemacht hat. Es waren ja teilweise doch eine ganze Menge.

* Kai Bolay
  Der die AMOK-Disketten fleißig zusammenstellt, und dabei immer das Veröffentlicht hat was ich ihm geschickt habe.

* /Z-NETZ/RECHNER/SPRACHEN/OBERON
  Wo sich merkwürdigerweise zum größten Teil nur Amiga Oberon Programmierer aufhalten

* de.comp.sys.amiga.(misc|adcvocacy)
  Wo der wohl größte Sprachenkrieg in deutschen Landen ausgeführt wurde.

1.15 KitCat.guide/Geschichte

Die Enwicklungsgeschichte
****

Revision: 37.1 Version: 1.0 [awn] 17-May-1993

Funktioniert und läuft.

Revision: 37.2 Version: 1.0 [awn] 17-May-1993

- Fehler im erzeugten Module entfernt, die OpenCatalog öffnete nur Kataloge die auch übergeben wurde, die default Einstellung vom System wurde ignoriert, wenn man nichts übergab dann wurd immer die eingebaute Sprache genommen.

Revision: 37.3 Version: 1.1ß [awn] 23-May-1993

^ neue Befehlstemplate. Diese ist jetzt an 'CatComp' orientiert.
DESCRIPTOR/A,TRANSLATION,CTFILE/K,CATALOG/K,MODFILE/K,L=LANGUAGE/K,
BL=BUILTINLANGUAGE/K,VER=VERSION/K/N

```
  + Catalog Translation Datei kann man sich erstellen lassen.

------------------------------------------------------------------------
Revision: 37.4   Version: 1.1ß   [awn] 27-May-1993

  + Wenn man beim erstellen eine Catalog Translation eine ältere
    angibt, dann werden die Zeichenketten der alten Übernommen. Wobei
    die, die nicht mehr in der Catalog Description sind rausfliegen.

------------------------------------------------------------------------
Revision: 37.5   Version: 1.1ß   [awn] 28-May-1993

  ^ V39 Interfaces Switch eingebaut, dadurch eine neue Template
    DESCRIPTOR/A,TRANSLATION,CTFILE/K,CATALOG/K,MODFILE/K,L=LANGUAGE/K,
    BL=BUILTINLANGUAGE/K,VER=VERSION/K/N,V39/S

  + Es können jetzt zwei Version vom Oberon-Quelltext erzeugt werden
    einmal für die V38 Locale.mod und einmal für die V39 Locale.mod von
    [hG], denn dort sind ein paar Konstanten anders geschrieben.
    beide Versionen laufen mit der locale.library V38

------------------------------------------------------------------------
Revision: 37.6   Version: 1.1ß   [awn] 29-May-1993

  + Nun ist es möglich sich mit CATALOG=<filename> eine Catalog
    erzeugen zu lassen.

  ^ In einer Zeichenkette können jetzt die folgende Steuerzeichen verwendet
    werden

    Zeichen  Bedeutung
      \a    display beep                (ASCII 7)
      \b    backspace                   (ASCII 8)
      \c    control Sequence Introducer (ASCII 155)
      \e    escape                      (ASCII 27)
      \f    formfeed                    (ASCII 12)
      \n    newline                     (ASCII 10)
      \r    carriage return             (ASCII 13)
      \t    tab                         (ASCII 9)
      \v    vertical tab                (ASCII 11)
      \xNN  Zeichen mit dem ASCII Code von NN in Hexadezimal
      \NNN  Zeichen mit dem ASCII Code von NN in Octal

    Andere werden mit einer Fehlermeldungen abgeschmettert. Die Steuerzeichen
    werden auch für das Oberon Modul entsprechend umgewandelt, wobei dort
    noch " besonders behandelt werden.

  - Fehler beim Zusammenfügen der Translation-Dateien entfernt.

------------------------------------------------------------------------
Revision: 37.7   Version: 1.1ß   [awn] 03-Jun-1993

  - Es wurden die Strings falsch nummeriert (immer bei 1 angefangen) wenn
    bei ersten String ein (//) für die Stringnummer übergeben wurde.
    (Report: Frank Dürring)

------------------------------------------------------------------------
```

```
Revision: 37.8   Version: 1.1ß   [awn] 12-Jun-1993

  - Es musste nach den ## Zeilen in einer Übersetzungsdatei unbedingt eine
    Kommentar Zeile kommen, sonst wurden die Kataloge und die neuen
    Übersetzungsdatei falsch erstellt. Jetzt kann dort auch direkt
    ein Bezeichner stehen.

------------------------------------------------------------------------
Revision: 37.9   Version: 1.1ß   [awn] 15-Jun-1993

  + Es werden jetzt auch C-Quelltexte erzeugt
    (Quelltext von Jochen Wiedmann)
    Dadurch ergibt sich eine neue Template
    DESCRIPTOR/A,TRANSLATION,CTFILE/K,CATALOG/K,OFILE/K,CFILE/K,
    L=LANGUAGE/K,BL=BUILTINLANGUAGE/K,VER=VERSION/K/N,V39/S

    ein 'Cd2Oberon Cd2Oberon.cd CFILE=Cd2OberonLocale'
    erzeugt nun zwei C-Files.
    Ein mal "Cd2OberonLocale.h" mit allen #define der Strings
    und "Cd2OberonLocale.c" wo die Prozeduren zum öffnen und schliessen
    von Catalogen, auslesen der Strings vorhanden sind.

  ^ MODFILE/K wurde zu OFILE/K

  ^ Es wird jetzt beim Pfadabschneiden auch das ":" berücksichtigt.

------------------------------------------------------------------------
Revision: 37.10  Version: 1.1ß   [awn] 17-Jun-1993

  - Bugfixes im C-Quelltext,
    (Report: Jochen Wiedmann, von wem auch sonst)

------------------------------------------------------------------------
Revision: 37.11  Version: 1.1ß   [awn] 21-Jun-1993

  - Wenn Übersetze Strings in der übersetzungdatei fehlten, und man versuchte
    mit der alten eine neue Zu erstellen, dann waren plötzlich übersetze
    Strings drin ... leider waren es nur die nachfolgenden Bezeichner.

------------------------------------------------------------------------
Revision: 37.12  Version: 1.1ß   [awn] 23-Jun-1993

  ^ Die Übersetzungs- und Beschreibungsdatei werden nun auch auf Komplettheit
    geprüft (z.B. vorhanden sein der Bezeichner) wenn man sich einen C- bzw.
    Oberon-Quelltext erzeugt (natürlich nur wenn auch eine Übersetungsdatei
    angegeben worden ist).
    (Report: Jochen Wiedmann)

------------------------------------------------------------------------
Revision: 37.13  Version: 1.1ß   [awn] 23-Jun-1993

  + Hinter dem Bezeichner in der Beschreibungsdatei ist es jetzt auch möglich
    die Länge anzugeben. So ergibt sich folgender Aufbau.

    (num/min/max)
```

```
    Alle Parameter sind Optional. Wenn keine Längenangabe (min und max)  ↩
       vorhanden
    ist, dann wird die entsprechende länge nicht geprüft. Aber ein (//) muss
    mindestens vorhanden sein.

------------------------------------------------------------------------
Revision: 37.14  Version: 1.1ß   [awn] 25-Jun-1993

  ^ Die eingebauten Zeichenketten (Sprache), alle verbessert und  ↩
     vereinheitlich
    (Begriffsverwendung)

------------------------------------------------------------------------
Revision: 37.15  Version: 1.1ß   [awn] 25-Jun-1993

  ^ Unbenannt von Cd2Oberon nach KitCat, weil es ja mitlerweile ja mehr macht
    wie nur aus der Katalogbeschreibung einen Oberon-Quelltext zu erzeugen.

------------------------------------------------------------------------
Revision: 37.16  Version: 1.1ß   [awn] 25-Jun-1993

  ^ wenn der ## language bzw. ## version Befehl in der Katalogübersetzung
    fehlt, dann wird eine Fehlemeldung ausgeben.
    (Report: Jochen Wiedmann)

  ^ Es wird jetzt nicht bei jedem kleinen Fehler abgebrochen, sondern nur ein
    ACHTUNG! mit Beschreibung ausgeben und dann weiter fortgefahren, so  ↩
       können
    mehrere Fehler auf einmal angesehen werden. Aber bevor man den Katalog
    erzeugen läßt MÜSSEN alle ACHTUNG! Meldungen verschwunden sein.
    (Vorschlag: Jochen Wiedmann)

------------------------------------------------------------------------
Revision: 37.17  Version: 1.1ß   [awn] 05-Jul-1993

  - Es wurde beim Oberon Quelltext beim öffnen das Kataloges, wenn man eine
    Sprache explizit angeben hat, die Versionummer des zu öffnenden Katalogs
    überschrieben.
    (Report: Thomas Igracki)

------------------------------------------------------------------------
Revision: 37.18  Version: 1.1ß   [awn] 06-Jul-1993

  + Neuer Parameter beim Programmstart: BASENAME; somit auch eine neue
    Template.

    DESCRIPTOR/A,TRANSLATION,CTFILE/K,CATALOG/K,OFILE/K,CFILE/K,
    L=LANGUAGE/K,BN=BASENAME/K,BL=BUILTINLANGUAGE/K,VER=VERSION/K/N,V39/S

    Bei BASENAME kann man angeben ob der Katalog noch in einem Speziellen
    Pfad liegt z.B: "LOCALE:Catalogs/deutsch/Tools/KitCat.catalog"
    Wenn man hier also als BASENAME = "Tools" übergibt so wird dies beim
    zu öffnenden Katalogs in den erzeugten Quelltexten vorgesetzt.
    So das der Katalog in dem Unterverzeichnis zu finden ist.
    (Vorschlag: Frank Dürring)

------------------------------------------------------------------------
```

```
Revision: 37.19  Version: 1.1ß   [awn] 06-Jul-1993

  - Ein Dos.IoErr() sollte jetzt nur noch am Ende des Programm ausgegeben
    weden wenn wirklich ein Dos-Fehler war. Es waren ab und zumal sinnlose
    Meldungen zu sehen.

------------------------------------------------------------------------
Revision: 37.20  Version: 1.1ß   [awn] 09-Jul-1993

  - Wenn eine Zeichenkette in der Beschreibungsdatei zu lang war, dann
    brach das Programm mit einem NilChk ab. Naja, jetzt nicht mehr.
    (Report: Jochen Wiedmann)

  - Bei der Ausgabe der Fehlermeldung ob eine Zeichekette zu lang oder
    zu kurz ist, wurde Bezeichner und Datei-Name vertauscht.

------------------------------------------------------------------------
Revision: 37.21  Version: 1.1ß   [awn] 09-Jul-1993

  + ENV-Variablen Unterstützung, bevor jetzt die Übergabeparameter
    ausgewertet werden, werden zuerst die Parameter aus der Datei
    'ENV:KitCat' ausgewertet. So kann man hier ein paar Standardparamter
    (wie z.B. V39) eintragen und dies brauch man dann nicht immer
    beim Aufruf anzugeben.
    Allerdings hat die Kommandoszeile die höhere Priorität gegenüber der
    ENV-Variable.

  ^ Die Template wurde wegen o.g. Funktion erweitert.

    DESCRIPTOR/A,TRANSLATION,CTFILE/K,CATALOG/K,OFILE/K,CFILE/K,
    L=LANGUAGE/K,BN=BASENAME/K,BL=BUILTINLANGUAGE/K,VER=VERSION/K/N,
    V39/S,V38/S

    Mit V38 kann man dann noch Explizit den V38 Oberon-Quelltext
    erzeugen lassen, falls man in der ENV-Variable V39 stehen hat.

------------------------------------------------------------------------
Revision: 37.22  Version: 1.1ß   [awn] 09-Jul-1993

  + Neue Funktion: Der Dupecheck, überprüft die Katalogbeschreibung auf
    doppelte Bezeichner. Dadurch ergibt sich auch eine neue Template

    DESCRIPTOR/A,TRANSLATION,CTFILE/K,CATALOG/K,OFILE/K,CFILE/K,
    L=LANGUAGE/K,BN=BASENAME/K,BL=BUILTINLANGUAGE/K,VER=VERSION/K/N,
    V39/S,V38/S,DUPECHECK/S,NODUPECHECK/S

    DUPECHECK schaltet ihn ein, und NODUPECHECK schaltet ins aus, falls
    er in der ENV-Variable fest eingestellt wurde.
    (Vorschlag: Jochen Wiedmann)

------------------------------------------------------------------------
Revision: 37.23  Version: 1.1ß   [awn] 09-Jul-1993

  - In dem Oberon-Quelltext wurden für die " als "" anstatt \" eingetragen
    so, das sie im Ausführbaren Programm nicht vorhanden waren.

------------------------------------------------------------------------
```

```
Revision: 37.24  Version: 1.1ß   [awn] 10-Jul-1993

  + Neue Funktion: Das Erzeugen von den Zeichenketten bei den Quelltexten ↩
     kann
    nun so eingestellt werden das mehrzeilige Zeichenketten die man in der
    .cd Datei definiert auch so im Quelltext stehen, und nicht in einer
    Zeile, nur beim Oberon-Quelltext mit auch eine Zeile direkt hinter der
    Bezeichner Konstante sein (Compilerfehler?), wie es bei C-Source
    aussieht kann ich nicht sagen ... mangels Testmöglichkeiten.

    Deswegen gibt es auch eine neue Template:

    DESCRIPTOR/A,TRANSLATION,CTFILE/K,CATALOG/K,OFILE/K,CFILE/K,
    L=LANGUAGE/K,BN=BASENAME/K,BL=BUILTINLANGUAGE/K,VER=VERSION/K/N,
    V39/S,V38/S,DC=DUPECHECK/S,NDC=NODUPECHECK/S,SS=SHORTSTRINGS/S,
    NSS=NOSHORTSTRINGS/S

    Wohl selbsterklärend.
    (Vorschlag: Frank Dürring und Jochen Wiedmann)

------------------------------------------------------------------------
Revision: 37.25  Version: 1.1ß   [awn] 10-Jul-1993

  + Jetzt werden die Kommentare die man in der Katalogübersetzungs-Datei
    eingeben hat auch mit übernommen, allerdings nur die Sachen die
    vor einem Bezeichner stehen und dann auch nur die Kommentare deren
    Zeile NICHT mit einem doppelten Semikolon beginnen (die ab dieser
    Version vom Programm erzeugten Kommentare werden so gekennzeichnet
    und die sollen ja nicht immer und immer wieder mit übernommen werden,
    denn dann würden sie sich vervielfachen.
    So sollte man bevor mal alte Katalogübersetzungen hat entweder die
    Semikolons von Hand verdoppeln, oder einfach kurz duch KitCat
    jagen.
    (Vorschlag: Jochen Wiedmann)

------------------------------------------------------------------------
Revision: 37.26  Version: 1.1ß   [awn] 12-Jul-1993

  + Quelltext Erweiterung. Beim Oberon Quelltext werden jetzt die
    Konstanten "numStrings", "minId" und "maxId" exportiert. Naja, numStrings
    gibt die Anzahl der Zeichenketten an die im definiert sind, minId ist
    die kleinste Id Nummer und maxID die höchste Id Nummer.
    Im C-Quelltext werden sie als num%sStrings min%sId und max%sId definiert.
    Wobei %s durch den Entsprechenden Programmnamen ersetzt wird (wie auch
    bei Open%sCatalog())
    (Dies ist für Thomas Igracki eingebaut worden damit er damit etwas
    machen kann, was ich nicht in die erzeugten Quelltexte einbauen wollte,
    das es zu spezifisch war, so wurde halt ein Kompromiß gefunden).

------------------------------------------------------------------------
Revision: 37.27  Version: 1.1ß   [awn] 12-Jul-1993

  - DUPECHECK Optionen bei den Programmparamtern entfernt, es wird nun
    immer auf doppelte Bezeichner geprüft. Deshalb wieder eine neue
    Template.

    DESCRIPTOR/A,TRANSLATION,CTFILE/K,CATALOG/K,OFILE/K,CFILE/K,
```

```
    L=LANGUAGE/K,BN=BASENAME/K,BL=BUILTINLANGUAGE/K,VER=VERSION/K/N,
    V39/S,V38/S,SS=SHORTSTRINGS/S,NSS=NOSHORTSTRINGS/S

------------------------------------------------------------------------
Revision: 37.28  Version: 1.1ß   [awn] 12-Jul-1993

  ^ Viele Interne Änderungen.

  ^ Das erstellen von Dateien ist jetzt schneller geworden.

------------------------------------------------------------------------
Revision: 37.29  Version: 1.1ß   [awn] 13-Jul-1993

  - Es konnte bei mehrzeiligen Zeichenketten zu ein Bereichsfehler
    kommen, jetzt nicht mehr.

  ^ Der Oberon Quelltext; Bei OpenCatalog() ist es jetzt auch möglich
    noch eine geöffnete LocalBase zu übergeben, oder NIL wenn man es
    nicht will. So ist es auch hier wie beim C-Quelltext und auch ein
    bisschen Flexibler.

  ^ Bei der Überprüfung auf doppelte Bezeichner wird nun nicht mehr
    wie wild Ausgaben gemacht, wenn die Bezeichner mehr als zweimal
    drin sind.

  - Überflüssige Includes aus dem C-Quelltext entfernt
    (Report: Jochen Wiedmann)

  ^ Nicht mehr ganz so Speicherfressend; beim Einlesen der Dateien wird
    kein Überflüssiger Speicher mehr reserviert.

  ^ Die eingebauten Zeichenketten etwas unformtiert so, das man nicht
    unbedingt mindestens 100 Zeichen pro Zeile braucht (aber gegen
    megalange Dateinamen werde ich nichts machen, das lohnt den
    Aufwand nicht).

------------------------------------------------------------------------
Revision: 37.30  Version: 1.1ß   [awn] 13-Jul-1993

  ^ Es wird jetzt an anderen Zeitpunkten die Dateien überprüft, so wird
    jetzt z.B. auch die Katalogbeschreibung überprüft wenn keine
    Katalogübersetzung vorhanden ist.
    (Report: Jochen Wiedmann)

------------------------------------------------------------------------
Revision: 37.31  Version: 1.1ß   [awn] 14-Jul-1993

  ^ Neue Variante für den C-Quelltext.
    Dadurch gibt es einen eine neue Befehlsschablone.

    DESCRIPTOR/A,TRANSLATION,CTFILE/K,CATALOG/K,OFILE/K,CFILE/K,
    L=LANGUAGE/K,BN=BASENAME/K,BL=BUILTINLANGUAGE/K,VER=VERSION/K/N,
    V39/S,V38/S,SS=SHORTSTRINGS/S,NSS=NOSHORTSTRINGS/S,OC=ONECAT/S,
    MC=MULTIPLECAT/S

    ONECAT
      Wenn man nur einen Sprachkatalog im Programm hat sie wird nur ein
```

```
      GetString() und nicht ein GetFooBarString() im Quelltext erstellt,
      so das man sich dan nicht immer die Finger wundtippen muss.
      Dies wird nur bei GetString() gemacht, ein OpenFooBarCatalog() muss
      weiterhin gemacht werden.

    MULTIPLECAT
      Schaltet das GetFooBarString() wieder ein. Falls man mehrere
      Sprachkatalog verwenden in einem Programm verwenden möchtte oder
      muss.

    Dies Bezieht sich nur auf den C-Quelltext, nicht auf den Oberon-Quelltext
    dort ist das, Sprachbedingt, einfacher gelöst.

-------------------------------------------------------------------------
Revision: 37.32  Version: 1.1ß   [awn] 14-Jul-1993

  - Es wurde in den Quelltexten, und Sprachkatalogen alle Zeichenketten mit ↩
     dem
    gleichen Inhalt generiert.

  - Die Programmparamter BASENAME, ONECAT und MULTIPLECAT sind weggefallen
    Also gibt es wieder neue Paramter.

    DESCRIPTOR,TRANSLATION,CTFILE/K,CATALOG/K,OFILE/K,CFILE/K,L=LANGUAGE/K,
    BL=BUILTINLANGUAGE/K,VER=VERSION/K/N,V39/S,V38/S,SS=SHORTSTRINGS/S,
    NSS=NOSHORTSTRING/S:

    Damit keine Funktionen verloren gehen, gibt es jetzt, wie bei CatComp ↩
       auch
    Steuerbefehle in der Katalogbeschreibung, Steuerbefehle beginnen mit
    einem # direkt gefolgt von dem Befehlsnamen, ohne Leerstelle. Folgende
    #-Befehle werden bis jetzt unterstützt

    #header <headername>
      Nur C-Quelltext: Hier kann man angeben ob man eine eigene Definition ↩
         für
      die .H Datei anlegen will. (Die Abfrage, ob die Datei schon da ist, ↩
         wird
      darüber geregelt.

    #catalogname <catalogname>
      Name des Sprachkatalog der geöffnet werden soll, allerdings ohne das
      .catalog (z.B. KitCat). So wird der Name der Katalobeschreibung ↩
         genommen,
      wie bisher.

    #function <funcname>
      Der Name der GetString() Prozedur, vereingestellt ist hier Get%sString ↩
         ()
      wobei an die stelle des %s der Katalogname, ohne .catalog gesetzt wird.
      Dadurch wird die ONECAT und MULTIPLECAT Option überflüssig.

    #basename
      Ersetzt den BASENAME Parameter beim Programmstart

    #array
```

```
      Hier kann der Name des Array welches die Zeichenketten beinhaltet  ↩
         definiert
      werden voreingestellt ist "KC%sArray", wobei %s durch den Katalognamen  ↩
         ersetzt
      wird. Nur beim C-Qulltext.

------------------------------------------------------------------------
Revision: 37.33  Version: 1.1ß   [awn] 15-Jul-1993

  ^ Oberon-Quelltext erweitert. Hier gibt es jetzt die Prozedur CloseCatalog ↩
     ()
    hiermit kann man den geöffnete Sprachkatalog wieder schliessen. Diese
    Prozedur wird von OpenCatalog() auch aufgerufen. So das man ihn vor
    einem OpenCatalog() aufruf nicht aufrufen muss.
    Hiermit ist es möglich auch die Sprache wärend des Programmablaufes
    zu ändern. Allerdings, müsste das Programm auch damit fertig werden und  ↩
       somit
    alles was irgendwie schon am Bildschirm zu sehen ist, neu aufbauen.

------------------------------------------------------------------------
Revision: 37.34  Version: 1.1ß   [awn] 15-Jul-1993

  - BUILTINLANGUAGE und VERSION beim Programmaufruf entfernt, wiederum neue
    Template

    DESCRIPTOR,TRANSLATION,CTFILE/K,CATALOG/K,OFILE/K,CFILE/K,L=LANGUAGE/K,
    V39/S,V38/S,SS=SHORTSTRINGS/S,NSS=NOSHORTSTRING/S:

  + zusätzliche Steuerbefehle

    #language <sprache>
      gleiche Funktion wie das ehemalige Builtinlanguage beim Programmstart

    #version <versionsnummer>
      gleich Funktion wie das ehemalige Version beim Programmstart

    #arrayopts <arraytyp>
      Das Array welches in dem C-Quelltext erzeugt wird, ist voreingestellt
      auf "static const", dies kann bei Bedarf geändert werden.

    #lengthbytes <n>
      Wenn hier eine Ziffer angegeben wird, dann wird vor der Zeichenkette
      die Länger derselbigen angegeben, in der n Bytes. GetString() liefert
      dann einen Zeiger auf diesen Länge, und nach n Bytes folgt dann der
      String.

  + Katalogoptimierung, falls in der Katalogbeschreibung und  ↩
     Katalogübersetzung
    bei Bezeichner die Zeichenkette gleich ist, dann werden diese  ↩
       Zeichenketten
    nicht mit in den Sprachkatalog mitaufgenommen.

------------------------------------------------------------------------
Revision: 37.35  Version: 1.1ß   [awn] 15-Jul-1993

  - Wenn in der Katalogbeschreibung Bezeichner vorhanden waren, die in der
    Katalogübersetzung noch nicht vorhanden waren, dann brach das Programm
```

```
    ab.

  ^ Wieder mal die Zeit und Art der Überprüfung der Katalogübersetzung und
    Katalogbeschreibung geändert. Dies geht jetzt wie folgt.

    Bei:

    'KitCat FooBar.cd FooBar.ct'

      Überprüfung der Katalogübersetzung und der Katalogbeschreibug.

    'Kitcat FooBar.cd CTFile FooBar.ct' oder
    'Kitcat FooBar.cd FooBar.ct CTFILE FooBar.ct'

      Nur Überprüfung der Katalogbeschreibung

    Andere Parameter wie Quelltexteerzeugung verändern die die
    überprüfung nicht.

  ^ Das Dateien einladen geht jetzt viel schneller, trotzdem sind noch
    weiterhin beliebig lange (klar, vom Speicherplatz begrentze) Zeilen
    bei den Dateien möglich.

------------------------------------------------------------------------
Revision: 37.36  Version: 1.1ß   [awn] 15-Jul-1993

  + Neuer Steuerbefehl für die Katalogbeschreibung

    #prototypes

      hier kann man eigenen Prototypes, anstatt der voreingestellten,
      angaben.

      Voreingestellt ist.

      #prototypes extern void %s(struct Locale *, STRPTR);\n\
      extern void %s(void);\n\
      extern STRPTR %s(LONG);

    Das erste %s wird duch 'Open%sCatalog', das zweite durch
    'Close%sCatalog', und das dritte durch den Prozedurenamen, der
    wenn man nichts anderes eingibt 'Get%sString' lautet. Wobei
    deren %s durch den Katalognamen "#catname" ersetzt wird.

    Also sollte man die Reihenfolge der Prototype definition
    nicht ändern, dies würde nur ab OS 2.1 mit den Positionsparametern
    bei den %s Formatzeichen funktionieren.

  + C-Quelltext an den GNU-C angepasst
    (Anpassung: Jochen Wiedmann)

------------------------------------------------------------------------
Revision: 37.37  Version: 1.1ß   [awn] 15-Jul-1993

  - Aufrung der Änderung in der überprüfung in 37.35 wurden war es
    nicht mehr möglich, die Katalogübersetzungen zusammenzufügen.
```

^ Es werden jetzt nicht mehr die Kommentare aus der Katalogübersetzung wieder in die Katalogübersetzung eingesetzt, sondern jetzt werden die Kommentare aus der Katalogbeschreibung in die Katalogübersetzung eingesetzt.
(Vorschlag: Jochen Wiedmann)

Revision: 37.38 Version: 1.1ß [awn] 01-Aug-1993

- Wenn jetzt ein Oberon-Quelltext erzeugt werden soll, das ist asl Funktionsname für die GetString-Prozedur "GetString" voreingestellt.

- Es jetzt nicht mehr erlaubt eine Katalogübersetzung und einen Sprachkatalog mit einem Befehlsaufruf zu erstellen.

Revision: 37.39 Version: 1.1ß [awn] 03-Sep-1993

- Kleinigkeiten am C-Quelltext geändert (Fehler bei Close%sCatalog()) und etwas erweitert.
(Report & Vorschlag: Jochen Wiedmann).

Revision: 37.40 Version: 1.1ß [awn] 09-Sep-1993

- Verschlimmerbesserten C-Quelltext aus der 37.39 nun besser gestaltet
(Report: Jochen Wiedmann)

Revision: 37.41 Version: 1.1 [awn] 13-Sep-1993 (* Release *)

- Noch einen kleinen Fehler in der C-Quelltext Erzeugung entfernt
(Report & Verursacher :): Jochen Wiedmann)

Revision: 37.42 Version: 1.1a [awn] 26-Sep-1993

- Wenn KitCat die Steuerzeichen bemängelte dann gab die Fehler-meldung nicht die entsprechende Zeile aus, sondern mal garnichts oder auch mal irgendeinen Dateinamen.
(Report: Sönke Brandt)

- Wenn man seine Anführungszeichen aus lauter gewohnheit mit \" eingeben hat, dann wurde dies von KitCat mit einer Fehlermeldung quittiert (Es ist bei KitCat nicht nötig die Anführungszeichen besonders zu behandeln).
(Report: Sönke Brandt)

- Wenn man in seine Zeichenketten, die man evtl. aus alten Oberon Quelltexten übernommen hat, immer noch ein \[als "Control sequence Indtoducer benutzt hat, dann wurde dies auch mit einer Fehlermeldung quittiert (man sollte eigentlich, so wie es in der Anleitung steht, ein \c nehmen), dies wird jetzt genauso behandelt wie \c.
(Report: Kai Bolay, nur weiss er nicht das es mir den über einen Umweg mitgeteilt hat)

```
  - Enforcerhit beseitigt wenn KitCat auf eine leere Zeile stieß.
    (Report: Kai Bolay)

-------------------------------------------------------------------------
Revision: 37.43  Version: 1.1b   [awn] 26-Sep-1993 (* Release *)

  - Es werden jetzt auch leere Zeilen in der Katalogübersetzung und
    Katalogbeschreibung zugelassen, wenn sie Syntanktisch erlaubt
    sind.
    (Report & Vorschlag: Kai Bolay)
```

1.16 KitCat.guide/Zukunft

```
Zukunft
*******

 Tja, was soll das Programm in Zukunft bieten. Naja, es werden keine
speziellen Sprachen mehr unterstützt werden, es wird eine Datei geben in
der die Quelltexte beschrieben werden und so kann KitCat an jede Sprache
angepasst werden. Auch wird es somit möglich sein die Quelltexte an seine
Bedürfnisse anzupassen.
```

1.17 KitCat.guide/Index

```
                Index
*****


                 Adresse des Autors
                                  Autor

                 Aufbau der der Katalogbeschreibung
                   Katalogbeschreibung

                 Aufbau der Katlogübersetzung
                        Kataloguebersetzung

                 Bug-Reports
                                           Autor

                 Copyright
                                             Wichtig!

                 Danksagungen
                                          Danksagungen

                 Entwicklung
                                           Geschichte
```