MakeCat

COLLABORATORS			
	TITLE : MakeCat		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		March 28, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

MakeCat

1.1 MakeCat.guide

CHOOSE A LANGUAGE / KIES EEN TAAL

English

Nederlands

1.2 MakeCat.guide English

Contents:

1.3 introduction

MakeCat

MakeCat is a little program that helps you to create locale catalogs. These catalogs can be used by your programs to support different languages. (Note: localisation was introduced to the Commodore Amiga with OS revision 38, that is Workbench 2.1. Therefore, you can only use the catalogs on systems that have installed the system software of version 2.1 or higher.)

MakeCat is a tool that can create a catalog with the data of a plain ASCII file, that can be edited in any editor. Catalog files are somewhat harder to edit, because they are in the IFF.CTLG format.

So MakeCat is a program for programmers who want to make use of the functions in the locale.library in their programs.
Also available is UnMakeCat. This program is the 'inverse' of MakeCat. It creates an ASCII file from a catalog file. This is easy if you want to make changes to an already existing catalog.

1.4 how to use makecat

MakeCat can only be used from the CLI (or the Shell). The command line template looks like this:

```
MakeCat FROM/A,CATALOG/A,BUFFERSIZE/N
```

for example:

```
MakeCat MyFile MyApplication/MyCatalog
```

The from-file is an ASCII file that you have previously created. It should be in a specific format in order to be interpreted by MakeCat.
The catalog-file is created by MakeCat. This file is created in the directory LOCALE:Catalogs/language. So if you want the catalog to be in a particular subdirectory, you should include the directoryname in the catalog-filename (see example above). You are recommended to use a subdirectory to keep the directory 'clean'. Name the subdirectory after the application. For example, all the system catalogs are placed in a subdir called 'Sys'. Note that this subdirectory is not created by MakeCat. If you specify a non-existing subdirectory, MakeCat will fail. This is also the case if you try to create a catalog for a language that does not have a subdirectory in LOCALE:Catalogs. Use the Workbench install program to install new languages to your system.
In BUFFERSIZE, you can specify the size of MakeCat's internal buffer. This buffer is used to read a single line from your 'from' file and thus sets the maximum number of characters that may be in one line. The default value is 1024.

If you do not get error messages from MakeCat, all went well and the catalog can be used by your programs. Look at the C-program example to see

how this can be done.
(Note: if you are experimentating with catalogs, you should flush your memory every time you have changed an existing catalog, or the system will keep using the old one instead. Flushing the memory can be done with the AmigaDOS command avail. Use 'avail flush'.)

1.5 how to use unmakecat

UnMakeCat does the opposite of what MakeCat does: it makes an ASCII file
out of a catalog.
The CLI template is:

```
UnMakeCat CATALOG/A,TO/A,LANGUAGE/A,BUFFERSIZE/N
```

for example:

```
UnMakeCat Sys/dos.catalog RAM:ASCII_file english
```

The catalog file must be placed in the directory Locale:Catalogs/language. This is why you need to specify the language on the command line, because it could be possible that a certain catalog name exists in more than one directory. (Note that the language name should be written as it is in that language. So english is 'english', but german is 'deutsch'. The language name should also be written completely in lower-case, so not 'Français', but 'français'.)
The created ASCII file is then in the MakeCat format and could be transformed into a catalog immediately with

```
MakeCat
.
```

UnMakeCat is particulary useful for translating existing catalogs or to make changes to them. For example if the program PROG has a german catalog called PROG/Prog.catalog, you could make an ASCII file with UnMakeCat:

```
UnMakeCat PROG/prog.catalog RAM:ascii deutsch
```

Then you could change the language name on the first line of the ASCII file, for example to english and translate all the strings in the file to english. Then you could create an english catalog file with:

```
MakeDir Locale:Catalogs/english/PROG
MakeCat RAM:ascii PROG/prog.catalog
```

1.6 format specifications

In order to create a catalog file, MakeCat needs to have data to put into that file. The catalog data consists of strings, with an associated identification number. This number can be any long integer. If you want to read a string from the catalog later, you need to pass this id number to the locale functions.
Also, every catalog file must have a version number. The locale.library reads this number and a programmer could request to open a catalog only if

it has a certain number.

The format of the ASCII datafile is defined as follows:

line 1: Language
line 2: Version string
next lines: ID + string

The language is just the name of the language the catalog is supposed to be in. Note that is should be written in lower case completely.
The version string is the standard AmigaDOS version string. For those not familiar with it, it looks like this:

```
   $VER: name.catalog vv.rr (dd.mm.yy)
```

The "$VER:" is an identification string. The version and revision numbers are defined as vv en rr respectively (vv and rr may consist out of more or less than two digits). The part between parentheses defines the date at which your file was created, f.e. (21.1.94) means the 21st of january, 1994.
With the AmigaDOS command 'version' you can inspect the version number of all files that have a version string.

The next lines consist of only numbers and strings. A number is written in normal decimal notation. A string is written in the same way as you would do in C source code: placed between quotation marks. Special codes are preceeded by a backslash (\), for example \n is the code for a newline symbol. References to variables (%d, %s, %p etc.) can also be included (only if the strings are used with a C printf() like function, else it makes no sense).
Some examples of valid lines are:

```
0 "Any ID is allowed, even zero!\n"
1 "An integer value is later filled in here %d by printf()"
123456 "Any long integer value can be an ID\n"
4      "Order of the ID's is not important"
12 "Beep\aBeep\aBeep\a"
```

Note that every string should fit into one line. A line can contain as many characters as was given in the argument BUFFERSIZE minus one. If BUFFERSIZE was not specified, a line can contain 1023 characters.

If a line does not meet this format, an error message will be displayed and the rest of the line will be skipped. Note that empty lines also produce these messages, since they are not compliant with the format.

1.7 c-program example

```
/* Example C program to demonstrate how strings can be read from catalog
   files. This program opens a catalog called example.catalog and tries to
   get a string that has 1 as its ID. */


#include <stdio.h>
#include <exec/types.h>
```

```
#include <libraries/locale.h>
#include <clib/exec_protos.h>
#include <clib/locale_protos.h>

void main(void)
{
   struct Library *LocaleBase;
   struct Locale  *locale;
   struct Catalog *catalog;
   long   StringNum;
   STRPTR String;

   /* To use the locale functions, we need to open locale.library */
   LocaleBase = OpenLibrary("locale.library", 38);
   if (LocaleBase != NULL)
   {

      /* We also need access to a data structure called locale */
      locale = OpenLocale(NULL);
      if (locale != NULL)
      {

         /* Now we specify that we want access to the strings in the
            catalog file "example.catalog". This catalog must first be
            created (with MakeCat for example). */
         catalog = OpenCatalog(locale, "example.catalog", TAG_END);
         if (catalog != NULL)
         {
            /* Specify that we want the string with 1 as ID. */
            StringNum = 1;

            /* Get the string from the catalog. If no string with the
               specified ID exists in the catalog, we will receive the
               string "Not in catalog." */
            String = GetCatalogStr(catalog, StringNum, "Not in catalog.");

            /* Print the string we just got to stdout. */
            printf("String[%d] = '%s'.\n", StringNum, String);

            /* Always close what you opened. */
            CloseCatalog(catalog);
         }
         else printf("Could not open catalog.\n");
         CloseLocale(locale);
      }
      CloseLibrary(LocaleBase);
   }
   else printf("locale.library version 38 or higher is required.\n");
}
```

1.8 bug reports etc.

If you find any bugs in MakeCat or it does not seem to work under specific circumstances, or if you want to see some improvements to it, send me a letter stating the error or your request along with a description of your

system setup and the circumstances under which the error occurs and I will
try to fix the bug.

My address is:

Camiel Rouweler
Oranjestraat 9
5611 JG Eindhoven
the Netherlands

1.9 program history

31-jan-94 First complete working version.
      (v38.00)

1.10 disclaimer

MakeCat is public domain and may be used and copied freely, if you do not
deviate from the following rules:

1) The author of MakeCat can not be held liable for any damage or harm
   that is the result of the use of MakeCat. Also I can not guarantee
   that MakeCat is free of errors or even virusses. All responsibility
   is for the user of this software.

2) MakeCat can be distributed freely, as long as nobody is making
   profit from my work. I am not making any profit, so it would not be
   fair to make money with the work of somebody else.
   Only a small fee for copying may be asked. This fee may not exceed
   $5,- or the equivalent amount in other valuta.

3) MakeCat may be not included in any public domain library without
   the written permission of the author. I herby permit Fred Fish to
   include MakeCat in the AmigaLibDisk library.
   (It is very unlikely I should refuse somebody to include MakeCat in
   his or her library, but I want to know where it is included.)

4) When distributing MakeCat, it should be distributed as it is now.
   No changes should be made to the binary or the other files. Also
   all files in the Makecat directory should be distributed with it
   and no other files should be added.
   An exeception is that if you want to add a translation of this
   manual, you are permitted to do so.

1.11 MakeCat.guide Nederlands

Contents:

1.12 introductie

MakeCat

MakeCat is een klein programma waarmee gemakkelijk gelokaliseerde catalogs gemaakt kunnen worden. Deze catalogs kunnen door jou programma's gebruikt worden om verschillende talen te ondersteunen.
(Opmerking: lokalisering werd bij de Commodore Amiga geïntroduceerd met versie 38 van de systeemsoftware, dat is Workbench 2.1. Het gebruik van catalogs en de locale.library is daarom voorbehouden aan systemen die deze software of een nieuwere versie geïnstalleerd hebben.)

MakeCat is een hulpprogramma dat met de data in een gewoon ASCII bestand de door de locale.library gebruikte catalog files maakt. De ASCII bestanden kunnen met praktisch iedere tekst editor geëdit worden. Dit is niet waar voor de catalog files, die in een special IFF (CTLG) formaat staan.

MakeCat is dus een programma voor programmeurs die hun programma's meerwaarde willen geven door het ondersteunen van meerdere talen.
Ook verkrijgbaar is UnMakecat, dat precies het tegenovergestelde doet wat MakeCat doet: het maakt een ASCII bestand van een catalog bestand. Dit is vooral geschikt om bestaande catalogs te vertalen of om daarin veranderingen aan te brengen.

1.13 gebruik van makecat

MakeCat kan alleen van de CLI (of de Shell) aangeroepen worden. Het CLI sjabloon ziet er zo uit:

```
MakeCat FROM/A,CATALOG/A,BUFFERSIZE/N
```

Bijvoorbeeld:

```
MakeCat MijnBestand MijnProgramma/MijnCatalog
```

Het 'from' bestand is een ASCII bestand dat je zelf eerder gemaakt hebt.

Het moet in een speciaal
formaat
gezet zijn, zodat MakeCat het kan
interpreteren. Het 'catalog' bestand wordt door MakeCat aangemaakt. Dit
bestand wordt in de directory LOCALE:Catalogs/taal geplaatst. Als je je
catalog in een bepaalde subdirectory hiervan wilt hebben, moet je die
subdirectory ook aangeven bij de catalog naam (zie voorbeeld hierboven).
je wordt aangeraden om subdirectory's te gebruiken, zodat de
'hoofddirectory' schoon blijft. Gebruik bij voorkeur de naam van je
toepassing als naam voor de subdirectory. Alle systeembestanden
bijvoorbeeld zijn geplaatst in de subdirectory 'Sys'. Merk overigens op dat
deze subdirectory niet door MakeCat gecreëerd wordt. Als je een
subdirectory aangeeft die niet bestaat, zal MakeCat zichzelf afbreken. Dit
gebeurt ook wanneer je een taal opgeeft waarvoor nog geen subdirectory in
LOCALE:Catalogs gecreëerd is. Gebruik het installatieprogramma op de
Workbench Install disk om nieuwe talen op je systeem te installeren.
BUFFERSIZE geeft de groote van de interne buffer die MakeCat gebruikt. Dit
komt overeen met het maximaal aantal karakters op één regel. Als je lange
teksten in een catalog wil zetten, geef hier dan een hoge waarde op.
De standaardwaarde is 1024.

Als MakeCat geen foutboodschappen geeft, is alles goed gegaan en kan de
catalog gebruikt worden door je programma's. Zie het
C-programma voorbeeld
voor meer informatie over hoe dit gedaan kan worden.
(Opmerking: als je aan het experimenteren bent met nieuwe catalogs, moet je
het geheugen 'flushen' iedere keer dat je een bestaande catalog veranderd
hebt, of het systeem zal de oude blijven gebruiken. Het geheugen 'flushen'
kan gedaan worden met het AmigaDOS commando avail. Gebruik 'avail flush'.)

1.14 gebruik van unmakecat

UnMakeCat doet het tegenovergestelde wat MakeCat doet: het zet een ↩
catalog
bestand om in een ASCII bestand.
Het kommando regel sjabloon is:

```
UnMakeCat CATALOG/A,TO/A,LANGUAGE/A,BUFFERSIZE/A
```

bijvoorbeeld:

```
UnMakeCat Sys/dos.catalog RAM:ASCII_bestand nederlands
```

Het catalog bestand moet in de lade Locale:Catalogs/taal staan. De taal
moet geheel in kleine letters opgegeven worden en zo geschreven worden als
in die taal gebruikelijk is, dus Engels wordt 'english', Duits wordt
'deutsch' en Nederlands wordt 'nederlands' (aanhaaltekens niet invoeren).
Het gekreëerde ASCII bestand is in het MakeCat formaat en zou direkt weer
omgezet kunnen worden in een catalog bestand met
MakeCat
.

Het nut van UnMakeCat ligt vooral in het vertalen of aanpassen van
bestaande catalogs. Stel je hebt een programma PROG dat een Duits catalog
bestand heeft in Locale:Catalogs/deutsch/PROG/prog.catalog. Dat kan dan

omgezet worden in een ASCII bestand met:

```
UnMakeCat PROG/prog.catalog RAM:ascii deutsch
```

Met een editor kunnen dan alle strings vertaald worden, bijvoorbeeld naar het nederlands. Ook de eerste regel, waar 'deutsch' staat, moet veranderd worden in 'nederlands'. Het ASCII bestand kan dan naar een nederlandse catalog vertaald worden met:

```
MakeDir Locale:Catalogs/nederlands/PROG
MakeCat RAM:ascii PROG/prog.catalog
```

1.15 formaat specificaties

Om een catalog bestand aan te kunnen maken, heeft MakeCat gegevens nodig om in dat bestand te zetten. De catalog gegevens bestaan uit strings, met een bijbehorend identificatienummer. Dit nummer mag iedere long integer zijn. Als je later een string uit de catalog wil lezen, dan zul je dit ID nummer door moeten geven aan de functies van de locale.library.
Bovendien moet ieder catalog bestand een versienummer hebben. De locale bibliotheek leest dit nummer in en een programmeur kan vragen een bepaalde catalog slechts dan te openen, als de versie recent genoeg is.

Het formaat van het ASCII databestand is:

```
regel 1:            Taal
regel 2:            Versiestring
volgende regels:    ID + string
```

De taal is gewoon de naam van de taal waarin de catalog egschreven is. Voor de naam moeten wel alleen kleine letters gebruikt worden!
De versiestring is de standaard AmigaDOS versiestring. Voor hen die niet bekend zijn met de versiestring, hij is als volgt gedefinieerd:

```
   $VER: naam.catalog vv.rr (dd.mm.jj)
```

Waar '$VER:' geldt als identificatie van de versiestring. de versie- en revisienummers worden weergegeven door rr en vv respectievelijk (vv en rr mogen uit meer of minder dan twee getallen bestaan). Het gedeelte tussen haakjes definieert de datum waarop het bestand gemaakt of voor het laatst veranderd is; (21.1.94) bijvoorbeeld betekent 21 januari 1994.
Met het AmigaDOS commando 'version' kan je het versienummer van alle bestanden die een versiestring bezitten bekijken.

De volgende regels bestaan uit louter identificatienummers en strings. Een identificatienummer wordt in normale decimale notatie geschreven. Een string wordt geschreven zoals in C broncode: tussen aanhaaltekens (").
Speciale tekens in een string worden voorafgegaan door een backslash (\).
Bijvoorbeeld, \n is het teken voor een overgang op een nieuwe regel.
Refererenties naar nog in te vullen variabelen (%d, %s, %p etc.) kunnen ook opgenomen worden in een string. Dit heeft natuurlijk slechts dan zin, als de string later gebruikt wordt in een functie als printf() in C.
Voorbeelden van geldige regels zijn:

```
0 "Ieder ID is geldig, ook nul!\n"
```

```
1 "Hier %d wordt door printf() nog een integer ingevuld!"
123456 "Iedere long integer kan een idetificatienummer zijn"
4    "Volgorde van de ID nummers is niet belangrijk"
12 "Piep\aPiep\aPiep\a\n"
```

Merk op dat iedere string begrensd is tot maximaal één regel. Een regel mag evenveel tekens bevatten als wat opgegeven is in het argument BUFFERSIZE minus één. Als deze niet opgegeven is, geldt een maximale lengte van 1023 tekens

Als een regel in het ASCII bestand niet aan het beschreven formaat voldoet, wordt een foutboodschap weergegeven en de desbetreffende regel wordt overgeslagen. Merk op dat lege regels ook foutboodschappen veroorzaken, omdat ze immers niet aan het formaat voldoen.

1.16 c-programma voorbeeld

```
/* Voorbeeld C programma dat demonstreert hoe strings uit de catalog
   bestanden gelezen kunnen worden. Dit programma opent een catalog
   "example.catalog" genaamd en probeert een string met als identificatie 1
   terug te krijgen. */


#include <stdio.h>
#include <exec/types.h>
#include <libraries/locale.h>
#include <clib/exec_protos.h>
#include <clib/locale_protos.h>

void main(void)
{
   struct Library *LocaleBase;
   struct Locale  *locale;
   struct Catalog *catalog;
   long   StringNum;
   STRPTR String;

   /* Om de locale functies te mogen gebruiken, moeten we locale.library
      openen. */
   LocaleBase = OpenLibrary("locale.library", 38);
   if (LocaleBase != NULL)
   {

      /* We hebben ook toegang tot een 'locale' datastructuur nodig. */
      locale = OpenLocale(NULL);
      if (locale != NULL)
      {

         /* Nu geven we aan dat we toegang willen tot de strings in het
            catalog bestand "example.catalog". Deze catalog moet wel eerst
            worden gemaakt (bijvoorbeeld met MakeCat). */
         catalog = OpenCatalog(locale, "example.catalog", TAG_END);
         if (catalog != NULL)
         {
            /* Geef aan dat we de string met als ID 1 willen. */
```

```
            StringNum = 1;

            /* Lees de string van de catalog. Als geen string in de catalog
               het ID 1 heeft, krijgen we de string "Niet in catalog."
               terug. */
            String = GetCatalogStr(catalog, StringNum, "Niet in catalog.");

            /* De verkregen string naar stdout afdrukken. */
            printf("String[%d] = '%s'.\n", StringNum, String);

            /* Niet vergeten: altijd sluiten wat je geopend hebt. */
            CloseCatalog(catalog);
         }
         else printf("Catalog kon niet geopend worden.\n");
         CloseLocale(locale);
      }
      CloseLibrary(LocaleBase);
   }
   else printf("locale.library versie 38 of hoger is nodig.\n");
}
```

1.17 bug rapportages etc.

Als je bugs (fouten) in MakeCat vindt, of MakeCat (b)lijkt niet te werken onder bepaalde omstandigheden, of als je graag verbeteringen aan MakeCat zou zien, stuur me dan een brief waarin je de fout beschrijft (of je verzoek) samen met een uitgebreide beschrijven van je systeem opzet en eventueel de omstandigheden waaronder een fout optreedt. Ik zal dan proberen de fout te herstellen of de verbetering door te voeren.

Mijn adres is:

Camiel Rouweler
Oranjestraat 9
5611 JG Eindhoven
Nederland

1.18 programma geschiedenis

```
31-jan-94   Eerste geheel werkende versie.
      (v38.00)
```

1.19 vrijwaring

MakeCat is public domain en mag vrijelijk worden gebruikt en gekopieerd, mits niet van de volgende regels afgeweken wordt:

1) de auteur van MakeCat kan niet verantwoordelijk gesteld worden voor enige vorm van schade of leed, die het resultaat is van het gebruik van MakeCat. Ik kan ook niet garanderen dat MakeCat vrij van fouten

is en ook niet dat het kompleet virusvrij is. Iedere verantwoordelijkheid is voor de gebruiker van de software.

2) MakeCat kan vrij verspreid worden, zolang niemand er financieel beter van wordt. Ik zelf verdien niets op dit programma, dus het klopt niet als iemand profiteert van mijn werk.
Slechts een redelijk bedrag voor het kopiëren van diskettes mag gevraagd worden. Dit mag nooit hoger zijn dan fl 10,- of een equivalent bedrag in andere valuta.

3) MakeCat mag niet opgenomen worden in een public domain serie zonder de schriftelijke toestemming van de auteur. Fred Fish heeft hierbij toestemming om MakeCat op te nemen in de AmigaLibDisk bibliotheek. (Het is onwaarschijnlijk dat ik afwijzend reageer op een verzoek tot opname, maar ik wil wel eerst weten waar mijn software verspreid wordt.)

4) Als Makecat verspreid wordt, dan moet het verspreid worden zoals het nu is. Geen veranderingen mogen worden gemaakt aan de bestanden. Bovendien moeten alle bestanden samen verspreid worden en er mogen geen andere bestanden toegevoegd worden.
Een uitzondering op deze regel is dat als je een vertaling van deze handleiding toe wil voegen, dan mag dat.