ErrorMsg.Lisez-moi

COLLABORATORS			
	TITLE : ErrorMsg.Lisez-moi		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		January 6, 2023	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

ErrorMsg.Lisez-moi

1.1 Informations sur l'errormsg.library

```
                Objet: errormsg.library
~~~~~~

Version: 3.03
~~~~~~~~

Date: Mercredi 30 Novembre 1994
~~~~~

Auteur: Frédéric Delacroix
~~~~~~~

Status: Freeware
~~~~~~~
```

TABLE DES MATIERES

Contactez-moi

1.2 Distribution et disclaimer

Dans cette distribution, vous trouverez la version 3.03 de l'errormsg.library. Je la distribue en FREEWARE, ce qui signifie que n'importe qui est autorisé à la copier et à la distribuer tant que les conditions suivantes sont respectées:

- Tous les fichiers restent inchangés. Si vous avez des commentaires à faire, mettez-les dans un fichier séparé, et assurez-vous qu'il est clair que je n'en suis pas l'auteur. L'archivage est bien sûr autorisé.

- Tous les fichiers soient distribués ensemble. Cela inclut le fichier bibliothèque lui-même, le code glue, les fichiers includes, les autodocs, tous les fichiers des directories ErrorCom WhatError et Rexx, tous les icones, le script d'installation et ce fichier.

- Vous ne faites pas usage commercial de cette bibliothèque sans permission écrite de ma part. Mon adresse se trouve à la fin de ce document.

- Si vous êtes l'auteur d'un programme shareware, freeware, giftware, charityware, etc..., vous êtes autorisé à distribuer le fichier nommé errormsg.library avec tous ses catalogues et le script d'installation.

- Tous les programmes utilisant errormsg.library doivent avoir mention dans leur documentation du fait qu'errormsg.library est Copyright 1944 Frédéric Delacroix.

Errormsg.library est freeware, mais j'en conserve le copyright. Ce n'est pas du domaine public! Les collections comme les AmigaLib disks de Fred Fish, disks du CAM, sont autorisés à inclure errormsg.library dans leurs bibliothèques.

ErrorCom est FreeWare et ne peut pas être distribué indépendamment de errormsg.library. WhatError et son source sont dans le domaine public, vous pouvez en faire ce que vous voulez. Le script arexx nommé "ShowERM.rexx" est dans le domaine public, mais "ERMID.rexx" est FreeWare.

Bien sûr, je ne fais aucune garantie d'aucune sorte sur le fonctionnement correct de la bibliothèque ou des programmes associés. Vous l'utilisez entièrement à vos risques et périls, car bien que j'aie fait beaucoup de chasse aux erreurs, je ne peux pas être sûr qu'il n'y en a plus.

1.3 Installation de la bibliothèque

errormsg.library est une bibliothèque runtime. Pour qu'elle fonctionne correctement, ellet doit se trouver dans le directory qui est

assigné à LIBS: (généralement le directory Libs du disque de démarrage). Comme alternative, vous pouvez aussi utiliser un programme comme LoadLibrary pour une installation non définitive.

Comme errormsg.library est localisée, vous devez copier quelques fichiers catalogues dans votre directory LOCALE:Catalogs/<langue> si vous voulez qu'elle puisse marcher dans une autre langue que l'anglais. Vous y êtes obligé car une telle bibliothèque n'a pas d'assignement PROGDIR: pour y trouver les catalogues.

Pour plus de convivialité, je vous recommande d'installer le script nommé "ERMID.rexx" dans votre directory REXX:. L'installation de "ShowERM.rexx" est optionnelle.

Tout ce qui vient d'être dit est accompli par le script d'installation fourni. Double-cliquez son icône, et suivez les instructions.

1.4 Introduction

Tous les développeurs savent combien il est ennuyeux d'inclure dans leurs programmes des messages qui disent à l'utilisateur ce qui ne va pas en cas d'erreur. C'est une tâche longue et difficile pour le programmeur, qui préfèrerait faire des choses plus intéressantes.
De plus, inclure les messages d'erreurs dans l'exécutable augmente souvent de façon spectaculaire la taille des fichiers, et ces messages ne sont souvent qu'en anglais. D'où l'idée d'une bibliothèque partagée qui fournirait tous les messages d'erreur localisés que le système pourrait retourner. C'est exactement le rôle d'errormsg.library. Elle inclut une fonction pour retourner un pointeur sur le message à afficher, et des fonctions pour l'afficher. A partir de la V2.0, errormsg.library à une fonction d'interrogation qui lui permet d'être appelée depuis les programmes ARexx.

Dans l'errormsg.library, les messages d'erreur sont identifiés par un id système/sous-système (pour savoir de qui vient le message) et le code lui-même. La plupart des fonctions proposent des tags pour modifier le comportement de la bibliothèque. Voyez le fichier d'autodocs pour de plus amples informations.

1.5 Possibilités ARexx

A partir de la V2.0, errormsg.library peut être un hôte ARexx. Pour cela, vous devez la déclarer avec la commande RXLIB depuis le Shell:

```
RXLIB errormsg.library 0 -60 2
```

ou avec la fonction AddLib depuis ARexx:

```
call addlib('errormsg.library',0,-60,2)
```

Comme on peut le remarquer ci-dessus, l'offset d'interrogation pour

errormsg.library est -60. Il est différent de celui de l'amigaguide.library par exemple alors faites attention (une erreur et le gourou est là !:-). Réclamez aussi explicitement la version 2 de la bibliothèque pour que les utilisateurs de la version 1 ne voient pas le gourou quand ARexx essaie d'accéder à une fonction inexistante.

Il y a actuellement 5 fonctions ARexx implémentées; elles sont directement reliées aux fonctions normales de la bibliothèque, voyez les autodocs pour les détails.

Ces fonctions réclament des arguments, comme IDCMP, qui sont attendus sous forme numérique. Pour rendre l'écriture de programmes ARexx plus facile, j'ai ajouté un script d'aide. Il se nomme ERMID, et doit être appelé sous forme d'une fonction:

Nombre=ERMID(ID1,ID2,ID3,...) (jusqu'à 15 arguments)

Les arguments acceptés par ce script sont des mot-clés à traduire en un nombre, lequel peut alors être utilisé comme argument pour l'appel d'errormsg.library (il peut être utilisé pour autre chose bien sûr).

Les mot-clés actuellement reconnus sont tous les codes pour les systèmes et sous-systèmes de la bibliothèque, tous les attribus de la mémoire (pas les options) (utilisés par le système/sous-système ERMSYS_EXEC/ERMSUB_NoMemory) et les flags IDCMP.

Si plusieurs arguments sont donnés, les valeurs résultantes sont couplées par un or dans un résultat global. Ceci ne marche que pour les flags de la bibliothèque, les attributs de la mémoire et les flags IDCMP.

Pour avoir une idée claire, tapez cette ligne dans le shell, après avoir ajouté la bibliothèque à l'environnement ARexx (comme décrit ci-dessus), et le script ERMID.rexx dans votre directory REXX: :

```
rx "say displayerrormsg(ermid(MEMF_CHIP,MEMF_PUBLIC), ermid(ERMSYS_EXEC),
ermid(ERMSUB_NoMemory),'Ha|He|Hi|Ho|Hu','Titre',,
ermid(IDCMP_DISKINSERTED,IDCMP_DISKREMOVED))"
```

(ne tapez pas les retours chariots, tapez tout sur une ligne). Vous devriez alors avoir le message "Pas assez de mémoire CHIP" dans une fenêtre avec 5 gadgets, titrée "Titre", qui disparaîtra quand vous introduirez ou retirerez une disquette dans un lecteur.

Vous pouvez alors essayer le script de démonstration, nommé ShowERM.rexx, qui assez intuitif. Pour spécifier un code (pour l'erreur, le système ou le sous-système), vous pouvez soit choisir 1,2 ou 3 et entrer un nombre, soit choisir 1,2 ou 3, entrer un identificateur et choisir 4,5 ou 6.

1.6 Les fichier include

J'ai écrit les fichiers include standards _lib.i, .i et .h pour la bibliothèque, plus d'autres pour les prototypes et pragmas, en me basant sur ceux écrits par Nico François pour sa reqtools.library. Je n'ai pas pu les tester (protos et pragmas), alors si vous trouvez des bugs, corrigez

les fichiers et envoyez-les moi pour que je puisse inclure les bons dans la prochaine distribution.

Des fichiers include pour le pascal, oberon ou quoi que ce soit sont aussi les bienvenus.

1.7 Le code glue

Le code glue. Eh bien, tout ce que j'ai pu faire est écrire le code source pour les routines sas (stub), car je n'ai pas d'outil pour faire des bibliothèques convenables pour un compilateur C. Je suggère que quelqu'un les assemble en une bibliothèque linkable et me les envoie, pour que je puisse les distribuer avec la prochaine version. Les bons crédits seront bien sûr mentionnés... Merci!

Note: encore une fois, j'ai pris le code glue pour reqtools comme référence, merci à Nico François, auteur de cette merveilleuse bibliothèque.

1.8 ErrorCom et WhatError

Dans le directory ErrorCom, vous trouverez un programme nommé -surprise,surprise- ErrorCom (version 1.07). C'est une commodité qui vous permettra de voir tous les messages connus d'errormsg.library, en entrant les codes pour le système et le sous-système, et le code d'erreur dans une (belle) interface gadtools. Vous pourrez aussi voir les effets de DisplayErrorMsgA() et AlertErrorMsg(). Lancez simplement le programme pour une démonstration.

ErrorCom reconnaît deux mot-clés, qui peuvent être entrés soit sur la ligne de commande pour le CLI, ou comme tooltypes à partir du Workbench (note: le CLI requiert des guillemets autour de la description de la hotkey, le Workbench ne les aime pas). Ce sont CXPRI et SHOWWINKEY, respectivement pour régler la priorité du broker au sein de la liste de commodities, et pour indiquer la hotkey qui sera utilisée pour monter/cacher la fenêtre. Par défaut, la hotkey est "lcommand lshift `" et la priorité est 0. Il se pourrait que j'écrive une version MUI d'ErrorCom bientôt.

WhatError est une simple commande CLI qui vous permet d'afficher sur le canal de sortie standard (c'est-à-dire la fenêtre du CLI) un message d'erreur avec les arguments SYSTEM, CODE et SUBSYSTEM. Le source pour cette commande est fourni.

WhatError forcera le code à 0 si ERMSYS_EXEC/ERMSUB_NoLibrary est utilisé (nouveau pour la version 3.02), ErrorCom fera cela seulement pour un affichage ou une alerte.

1.9 Exemple de routine OpenLib

A partir de la version 3.02, errormsg.library inclut une paire système/sous-système nommée ERMSYS_EXEC/ERMSUB_NoLibrary. Elle est utilisée pour dire à l'utilisateur qu'une bibliothèque, un device ou une ressource a refusé de s'ouvrir. En utilisant un masque de flags (deux sont actuellement disponibles: EXECF_NOLIB_USENAME et EXECF_NOLIB_USEVERSION) dans l'argument Code, vous pouvez contrôler le niveau de détails du message produit: mettre le flag EXECF_NOLIB_USENAME retournera un message avec une place (code de formattage %s) pour le nom de la bibliothèque ou du device, et mettre les deux flags (n'utilisez pas EXECF_NOLIB_USEVERSION tout seul) retournera un message avec une place (code de formattage %ld) pour le numéro de version de la bibliothèque. Les ressources ne sont pas ouvertes avec un numéro de version, vous ne devez donc fournir un numéro de version dans le message d'erreur à moins que vous le testiez à la main (en espionnant la structure Resource). Attention, les ressources n'ont pas nécessairement besoin d'un numéro de version pour fonctionner (reportez-vous à leur documentation)!

Ces arguments supplémentaires peuvent être facilement utilisés par des fonctions à la RawDoFmt(), y compris EasyRequestArgs() et DisplayMessageA(). De plus, de façon à donner un accès totalement intégré, la fonction DisplayErrorMsgA() reconnaît deux nouveaux tags: EMT_LibName (ti_Data pointe sur le nom de la bibliothèque) et EMT_LibVersion (ti_Data est le numéro de version), pour fournir ces arguments directement.

Pour rendre cette possibilité facilement utilisable, je joins ici (en assembleur mais facilement traduisible en C) une routine nommée OpenLib(), dont le rôle est d'ouvrir la bibliothèque donnée en argument et d'alerter l'utilisateur si quelque chose ne va pas.

```
OpenLib ; (D0,Z)Base=OpenLib(LibName,Version)(A1,D0)
  movem.l d3-d4/a6,-(sp)
  move.l  a1,d3
  move.l  d0,d4
  move.l  Exec.Base(pc),a6
  jsr _LVOOpenLibrary(a6)
  tst.l d0
  bne.s .Good
  move.l  #ERMSYS_EXEC,d1
  move.l  #ERMSUB_NoLibrary,d2
  move.l  #EXECF_NOLIB_USENAME!EXECF_NOLIB_USEVERSION,d0
  clr.l -(sp)
  move.l  d3,-(sp)
  pea EMT_LibName
  move.l  d4,-(sp)
  pea EMT_LibVersion
  move.l  sp,a0
  move.l  _ErrorMsgBase(pc),a6
  jsr _LVODisplayErrorMsgA(a6)
  add.l #20,sp
  moveq #0,d0
.Good movem.l (sp)+,d3-d4/a6
  rts
```

Bien sûr, vous ne devez pas utiliser cette routine pour ouvrir l'errormsg.library elle-même. A la place, si errormsg.library refuse de s'ouvrir, vous devriez provoquer une alerte avec DisplayAlert(), disant que

votre application requiert errormsg.library. Alternativement, si vous ne voulez pas ouvrir Intuition vous-même, vous pouvez utiliser la fonction Alert() d'exec (avec pour paramètres AT_Recovery!AG_OpenLib)...

1.10 Comment traduire errormsg.library

Errormsg.library et ErrorCom sont complètement localisés, ce qui signifie que, pourvu que la locale.library soit disponible, ainsi que les fichiers catalogues, ils peuvent fonctionner dans votre langue.
Cependant, je ne connais pas d'autre langue que l'anglais et le français (j'ai oublié presque tout mon espagnol scolaire). Alors si vous voulez un catalogue pour votre langue, vous devrez traduire les chaînes vous-mêmes.

Pour cela, remplissez les blancs dans les fichiers de traduction de catalogues (ces fichiers qui se terminent par .ct) avec les traductions des chaînes qui sont en commentaire. Ensuite envoyez-moi le fichier résultant. Si tout va bien, vous recevrez bientôt le catalogue compilé. Ceci est valide pour errormsg.library et ErrorCom.

Les traductions des fichiers de documentation ou des scripts d'installation sont aussi les bienvenues.

Ooops. Presque oublié: dans le fichier de traduction pour ErrorCom, les étiquettes des gadgets doivent commencer par la lettre majuscule du raccourci clavier (elle n'est pas affichée), et le caractère à souligner doit être précédé de "_". Pour le gadget nommé "Message", mettez "\x01" comme premier caractère (pas de raccourci clavier).

1.11 Contactez-moi...

Je peux être contacté pour quoi que ce soit par courrier à:

Frédéric Delacroix
5 rue d'Artres
59269 QUERENAING, FRANCE.

Je salue tous mes amis, ProMedia, et AmigaNews.