AmiTex

FLORAC Roland

COLLABORATORS			
	TITLE : AmiTex		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	FLORAC Roland	August 19, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

AmiTex

1.1 Sommaire

```
                    ------------------------------------
AmiTex Version 3.00 1er janvier 1997
------------------------------------
```

1.2 Distribution

AmiTex n'appartient pas au domaine public, mais
peut être distribué par tous les moyens qui vous conviendront,
dans le respect strict des conditions suivantes :

- Gratuité :
La distribution devra être effectuée SANS CONTREPARTIE,
financière ou autre.
En particulier il ne pourra être proposé sur un serveur
recevant une partie du coût des communications, pas plus que
sur une disquette vendue plus cher que son prix d'achat.

- Intégralité :
Le programme doit être distribué accompagné de tous les
fichiers composant le package.

- Exceptions :

Roland Florac
, et lui seul, pourra exceptionnellement accorder
une permission de distribution dérogeant aux principes ci-dessus.
En particulier, l'inclusion d'AmiTex dans un ensemble
de logiciels distribués commercialement devra faire l'objet
d'accords écrits.
F.F.D. n'a pas le droit de distribuer ce logiciel.

- Utilisation :
Ce programme est libre (concept Freeware), vous pouvez l'utiliser
autant que bon vous semble pour une utilisation personnelle.

1.3 Présentation

AmiTex est un programme écrit en langage C (SAS C Compiler ↩
5.10a),
avec aussi un peu d'assembleur (Devpac 3.14).

C'est un éditeur de textes. Il possède une interface ARexx, avec
environ 100 commandes, avec la possibilité de définir ses propres
fonctions et d'utiliser des variables. Son fonctionnement requiert
le système 3.0 au moins. D'autre part certaines fonctions font appel
à la bibliothèque
bgui
.library version 39 ou plus. Celle-ci n'est
pas distribuée avec le logiciel, vous pouvez vous la procurer dans
le domaine public (site ou CR-ROM Aminet par exemple).

Ce programme permet de travailler sur un nombre de fenêtres
théoriquement illimité (sauf par la mémoire disponible !). Vous
pouvez bénéficier des fonctions avancées du système 3.0:
AppWindows

,

AppIcon
,
Pools
...

1.4 Installation

Le programme peut être copié dans tout répertoire, à votre convenance, sous réserve qu'il y ait assez de place libre.
J'utilise, pour ma part, le répertoire c:.
Le fichier de configuration Configuration.Amitex doit être copié dans le répertoire s:
Le fichier
d'aide
AmiTex.guide peut être copié dans tout emplacement à votre convenance, vous devrez cependant situer ce fichier à l'aide du tooltype
HELPFILE
de l'icône du programme
si vous voulez bénéficier de l'aide en ligne.

ATTENTION: le programme, pour fonctionner, fait appel aux ROM 3.0, son fonctionnement est impossible avec une version du système antérieure.

1.5 Lancement du programme / ToolTypes

La taille de la pile courante du CLI (ou du Shell) peut être fixée à 4 ko (suffisant).
AmiTex peut être lancé depuis le CLI ou le Workbench (en cliquant deux fois sur son icône).

Lancement depuis le CLI
AmiTex [Fichier1] [Fichier2]....
L'insertion de "run", permettant de rendre la main au Shell de suite, est inutile, le programme se détachant lui-même du Shell.
Les jokers AmigaDOS sont acceptés :
AmiTex #?.c ==> tous les fichiers comportant l'extension ".c" seront chargés chacun dans une fenêtre (à condition qu'il y ait suffisamment de mémoire bien sûr).
De même AmiTex #?.c #?.s (ou run AmiTex #?.[cs]) sera accepté.
Vous pouvez bien sûr lancer le programme sans spécifier de nom de fichier (le nom "Innomé" sera alors utilisé par défaut).

Lancement depuis le Workbench
Il suffit de cliquer deux fois sur l'icône du programme ou sur celle d'un texte possédant le nom du programme dans son champ TOOL-TYPES. Dans le premier cas vous pouvez utiliser la touche SHIFT pour sélectionner autant de textes que vous le désirez

avant de cliquer sir l'icône du programme.

L'icône du programme admet trois types d'outils:
- le type
WINDOW
,
- le type
HELPFILE
,
- le type
STARTUP
,
- le type
X_ICON
,
- le type
Y_ICON
.

Utilisez le menu Information du Workbench pour modifier ces types d'outils.

A noter que les requêtes qui apparaissent pour signaler un défaut ou une action peuvent être acquittées en cliquant sur le gadget VU à l'aide de la souris ou en frappant n'importe quelle touche au clavier (pratique...) ou même à l'aide du bouton droit de la souris.
De même les boîtes de requêtes proposant un choix (OUI ou NON) peuvent être acquittées de façon classique, à l'aide de la souris, ou bien à l'aide du clavier (ENTER ou RETURN pour OUI, ESC pour NON)...

La barre de titre des fenêtres de texte comportent certaines indications:
- le nom complet du fichier associé au texte,
- la mention " (modifié)" si le texte a été modifié et n'a pas été sauvé,
- une parenthèse (,
- le code du caractère courant, en hexadécimal (exemple: $41 pour un A),
- un espace,
- le numéro de colonne courant,
- un tiret,
- le numéro de ligne courant,
- une barre de fraction /,
- le nombre de lignes du texte,
- une parenthèse).

1.6 Type d'outil HELPFILE

Ce type d'outil permet de déterminer où se situe le fichier

d'aide
. Il doit être suivi du signe = puis du nom du fichier d'aide, avec son chemin complet. Vous pouvez ainsi placer ce fichier où vous le désirez, et même le renommer (cependant non conseillé !) puisque vous devez donner le nom complet du fichier.

Exemple:
HELPFILE=HELP:AmiTex.guide

1.7 Type d'outil WINDOW

Ce type d'outil permet de spécifier les dimensions de la fenêtre ouverte lors du démarrage du programme. Le mot WINDOW doit être suivi du signe = puis des coordonnées du coin supérieur gauche de la fenêtre et de sa largeur puis de sa hauteur. Il ne doit pas y avoir d'espaces entre les données.

À noter que les dimensions de la fenêtre sont sauvées avec l'icône du fichier si le menu
Préférences/Sauver icône
est
marqué. Ceci permet de retrouver les mêmes dimensions de la fenêtre lors d'un chargement de ce fichier par le Workbench ou en amenant cette icône sur l'icône d'application du programme, présente sur l'écran du Workbench.

Exemple:
WINDOW=0,1,1000,500 fixe l'emplacement de la fenêtre en haut à gauche de l'écran (il reste une ligne libre en haut pour descendre l'écran du Workbench). La largeur, fixée à 1000, prendra la largeur maximale admissible par l'écran du Workbench (640 pixels ou plus...) et la hauteur est fixée à 500 pixels (également si possible).

1.8 Type d'outil STARTUP

Ce type d'outil permet de lancer l'exécution d'un script ARexx
dès le démarrage du programme, ceci vous permet par exemple de définir
automatiquement les fonctions que vous souhaitez. Ce type doit être suivi du signe = puis du nom du script avec ou sans l'extension AmiTex.

Exemples:
STARTUP=REXX:startup.amitex
STARTUP=démarrage

1.9 Type d'outil X_ICON

Ce type d'outil permet de déterminer l'abcisse des coordonnées d'affichage de l'icône du programme sur l'écran du Workbench. Il suffit d'ajouter la ligne X_ICON=10 pour que l'icône soit affichée à gauche de l'écran du Workbench.

1.10 Type d'outil Y_ICON

Ce type d'outil permet de déterminer l'ordonnée des coordonnées
d'affichage de l'icône du programme sur l'écran du Workbench.
Il suffit d'ajouter la ligne Y_ICON=500 pour que l'icône soit
affichée en bas de l'écran du Workbench.

1.11 Les menus

Le programme AmiTex possède 5 menus, pouvant être sélectionnés à l'aide du bouton droit de la souris.

Menu Projet

Menu Édition

Menu Recherche

Menu Macros

Menu Préférences

1.12 Menu Projet

Ce menu possède 15 entrées:

Charger
Charge un texte dans la fenêtre active.

Insérer
Insérer un fichier texte dans la fenêtre active.

Sauver
Sauve le texte dans le fichier courant.

Sauver en
Sauve le texte en spécifiant un nom de fichier.

Renommer
Pour renommer la fenêtre active.

Annoter~fichier
Pour annoter un fichier.

Effacer~fichier
Permet de supprimer un fichier.

Réduire~fenêtre
Réduit la fenêtre active (icône).

Cacher~fenêtre

Ferme la fenêtre active, le texte reste en mémoire.

Autre~fenêtre
Ouvre une nouvelle fenêtre.

Nouveau
Vide la fenêtre courante.

Imprimer
Impression du texte contenu dans la fenêtre active

Informations
Affiche quelques informations.

Textes
Donne la liste des textes actuels.

Aide~spécifique
Affiche une aide AmigaGuide.

Quitter
Ferme tous les textes.

1.13 Menu Projet/Charger

Ce menu permet de choisir le texte à charger dans la fenêtre ↩ active.
La boîte de requête de fichiers utilise la bibliothèque asl.library (celle-ci est sur ~disque). À noter que celle-ci n'est chargée que lors de l'appel d'une des fonctions qui en a besoin (Chargement, Insertion, Sauver...). Vous pouvez cliquer deux fois sur le nom d'un fichier pour le sélectionner (ou sur VALIDE). La fenêtre prend le nom du fichier chargé. La boîte de requête peut être déplacée ou agrandie, à votre convenance, ses dimensions et sont emplacement seront conservés lors d'une utilisation ultérieure.
Si la fenêtre contenait un texte, celui-ci est perdu, sauf s'il avait été modifié mais non sauvé: le programme vous offre alors le choix de continuer l'opération de chargement (l'ancien texte est alors perdu) ou bien d'abandonner, vous permettant alors de le sauver.
Raccourci clavier : AMIGA-O (open) ou CTRL-O

Nota: vous pouvez également utiliser les fonctions du système 3.0 pour charger un fichier texte:
AppIcon
et
AppWindows
.

1.14 Menu Projet/Insérer

Ce menu permet d'insérer un fichier texte à l'emplacement où est situé le curseur (choix du fichier à l'aide du requester).

Si le curseur n'est pas situé en début de ligne, l'insertion
s'effectue quand même AVANT la ligne courante.
Raccourci clavier : AMIGA-E ou CTRL-E

Nota: vous pouvez également utiliser les fonctions du système 3.0
pour insérer un fichier dans une fenêtre (voir
AppWindows
).

1.15 Menu Projet/Sauver

SAUVER : sauvegarde du texte dans un fichier disque en utilisant
le nom de la fenêtre.
Raccourci clavier : AMIGA-S (sauver) ou CTRL-S

1.16 Menu Projet/Sauver en

Ce menu permet de sauvegarder le texte en choisissant le fichier
destination à l'aide d'une boîte de requête. Pour cette fonction
vous ne pouvez pas cliquer deux fois sur le nom d'un fichier pour
le sélectionner (obligation d'utiliser le gadget VALIDE pour
éviter les erreurs)
Si le nom de fichier choisi correspond à un fichier déjà existant
le programme vous demandera une confirmation avant de l'écraser.
Raccourci clavier : AMIGA-A (save As) ou CTRL-A

1.17 Menu Projet/Renommer

Ce menu permet permet de renommer le texte de la fenêtre en cours.
Le choix s'effectue à l'aide d'une boîte de requête de fichier.
Raccourci clavier : AMIGA-= ou CTRL-=

1.18 Menu Projet/Annoter fichier

Ce menu permet d'ajouter une note à un fichier choisi (effet
analogue à celui de la commande FileNote du DOS).
Cette note sera affichée lors de la commande List de l'AmigaDOS.
Il n'y a pas de raccourci clavier pour cette fonction.

1.19 Menu Projet/Effacer fichier

Ce menu permet, comme son nom l'indique, de supprimer un fichier...
Le fichier est choisi à l'aide du requester (pas de double clic là
non plus). Attention à ce que vous faites (il faut plus de temps
pour créer un fichier texte que pour l'effacer...).
Il n'y a pas de raccourci clavier pour cette fonction (danger...).

1.20 Menu Projet/Réduire fenêtre

Ce menu permet de réduire les dimensions de la fenêtre texte en cours. Une petite fenêtre apparaît alors en haut de l'écran avec le nom du texte seul.
Vous pouvez sélectionner cette fenêtre puis utiliser le bouton droit de la souris ou frapper une touche au clavier pour la faire revenir à son état initial. Cette fonction est bien utile quand on travaille sur de multiples fenêtres (il peut m'arriver d'en avoir jusqu'à quinze...)
Vous pouvez déplacer cette fenêtre a n'importe quel emplacement de votre choix, quand cette fonction sera rappelée ultérieurement, cette petite fenêtre reprendra cette place.

Raccourci clavier : AMIGA-I (icône) ou CTRL-I

1.21 Menu Projet/Cacher fenêtre

Ce menu permet de fermer la fenêtre active. Le texte reste cependant en mémoire. Pour faire réapparaître cette fenêtre vous pouvez cliquer deux fois sur l'icône d'application du programme, située sur l'écran du Workbench, ou bien utiliser un double clic sur le bouton droit de la souris, lequel vous permet d'afficher une liste de boutons contenant chacun le nom des diverses fenêtres présentes en mémoire.

1.22 Menu Projet/Autre fenêtre

Cette fonction permet l'ouverture d'une nouvelle fenêtre.
Une requête de fichier permet de choisir le texte à y charger (utilisez le gadget ANNULE si vous ne voulez rien y mettre, mais utilisez alors plutôt la touche F4, comme expliqué ci-dessous).

Raccourci clavier : touche de fonction F3.
La touche de fonction F4 permet la même chose mais sans provoquer l'ouverture de la requête de fichier.

Une nouvelle fenêtre peut être ouverte, en tapant sur F3 ou F4, même à partir d'une fenêtre iconifiée (cette dernière ne sera pas réouverte).

1.23 Menu Projet/Nouveau

Ce menu sollicite la demande d'effacement du texte présent dans la fenêtre (Attention...). Si le texte a été modifié, vous pourrez cependant le sauver (répondre Oui à la seconde requête).

Il n'y a pas de raccourci clavier pour cette fonction.

1.24 Menu Projet/Imprimer

```
                    Ce menu permet d'envoyer le texte à l'imprimante (périphérique ↩
                        PRT:).
  Le programme utilise les Préférences sélectionnées par le système:
  pilote d'imprimante, qualité, marges, ... Le texte entier est imprimé.
Raccourci clavier : AMIGA-P (Print) ou CTRL-P

  Voir aussi:
              PRINT
```

1.25 Menu Projet/Informations

```
                    Ce menu provoque l'affichage d'un certain nombre d' ↩
                       informations :
  copyright, nombre de caractères dans le texte, nom du
              port
                    ARexx, quantité de mémoire libre...
Raccourci clavier : AMIGA-K ou CTRL-K
```

1.26 Menu Projet/Textes

```
  Ce menu permet d'afficher dans une boîte le nom des différents textes
  et de leur nombre de lignes (10 textes au maximum). Le rang de la
  fenêtre (0 à 9) est affiché dans la colonne de gauche. Cette indication
  est très utile dans le cas où vous travaillez avec de multiples fenêtres:
  une touche ALT utilisée en conjonction avec l'une des touches
  0 à 9 du clavier numérique permet en effet de sélectionner la fenêtre
  correspondante. Ainsi la combinaison ALT-1 ramène la fenêtre de rang 1
  en avant-plan, celle-ci devenant la fenêtre active. Vous pouvez
  aussi faire défiler les fenêtres en utilisant les combinaisons ALT-+
  ou ALT-- (toujours en utilisant les touches + et - du clavier numérique).
  ALT-+ permet de passer à la fenêtre qui suit alors que ALT-- ramène la
  précédente. À noter l'indicateur + ou - signalant si le texte a été modifié.
  Enfin une dernière possibilité permet de changer de texte sans utiliser le
  clavier: un double clic sur le bouton droit de la souris amène une boîte
  de requête sur la fenêtre de travail, la liste des textes y figure (sans
  le chemin complet, seul figure le nom du fichier), il suffit alors de
  cliquer sur le bouton correspondant au nom du texte souhaité pour ramener
  sa fenêtre en avant plan.
Raccourci clavier : AMIGA-. ou CTRL-., double clic bouton droit ou
        combinaison ALT-N\textdegree{} fenêtre
```

1.27 Menu Projet/Aide spécifique

```
  Ce menu permet d'obtenir une aide à l'aide du programme AmigaGuide.
  Vous devez pour cela spécifier le nom d'un node appartenant au fichier
  AmiTex.guide. Ce fichier contient notamment des nodes pour chacunes des
  macros ARexx (Ex: COPY, PASTE...)
  À noter que vous pouvez également obtenir une aide à partir de chacune
  des entrées de menu en sélectionnant l'un de ces menus et en appuyant
  simultanément sur la touche HELP.
```

1.28 Menu Projet/Quitter

Ce menu permet de fermer toutes les fenêtres et de supprimer tous
les textes de la mémoire. Le programme quitte également en libérant
la mémoire qui'il occupait.
Si l'un des textes a été modifié, une sauvegarde sera proposée.
Raccourci clavier : AMIGA-Q (quitter)
CTRL-Q (ne ferme que la fenêtre active)

1.29 Icône d'application AmiTex

Le concept d'AppIcon (Icône d'application) autorise le ↩
chargement
d'un fichier de façon très simple, en une seule action, à l'aide
de la souris. Comme décrit, dans l'introduction, le programme crée
une icône qui apparaît sur l'écran du Workbench, lors de son

lancement
. Cette icône porte le nom d'AmiTex.
Il vous suffit de "saisir" une icône correspondant à un fichier
texte quelconque, à l'aide de la souris, et de la faire glisser
sur cette icône pour que ce fichier soit chargé dans une nouvelle
fenêtre.

Cette icône possède également une autre fonction: en cliquant deux
fois dessus rapidement, avec le bouton gauche, vous pouvez "réveiller"
le programme si toutes les fenêtres sont cachées (la fenêtre active
est réouverte), ou faire passer en avant-plan la fenêtre active si
celle-ci est cachée. Cette action peut également permettre un
déblocage éventuel, provoqué par une mauvaise utilisation de la
fonction
LOCK
dans un
script
ARexx.

1.30 Chargement de fichiers textes à l'aide des AppWindows

Toutes les fenêtres ouvertes en mode "normal" (c'est à dire ↩
qui ne
n'ont pas été
réduites
), sont des AppWindows.
Cette fonction est gérée par le système 3.0 et permet de détecter le
"largage" d'une icône sur ces fenêtres. La conséquence pour l'utilisateur
est que pour charger un texte dans une fenêtre, il suffit de cliquer
sur son icône avec le bouton gauche de la souris, puis de faire glisser
celle-ci sur cette fenêtre, en maintenant le bouton appuyé.
Attention: le nouveau texte sera INSÉRÉ dans la fenêtre si celle-ci
n'est pas vide. L'insertion s'effectuant à partir de la ligne sur
laquelle vous avez déposé l'icône. SI vous voulez charger ce texte
dans une nouvelle fenêtre, déposez l'icône sur l'

```
            AppIcon
             du programme,
présente sur l'écran du Workbench, ou ouvrez d'abord une
            nouvelle
                fenêtre (Appui sur F4 par exemple).
```

1.31 Gestion de la mémoire

```
Les pools correspondent à un concept de gestion de la mémoire
amélioré, permettant de ne pas trop fragmenter la mémoire. En
effet chaque ligne est mémorisée de façon dynamique, dans une
zone mémoire qui est allouée au programme au fur et à mesure
des besoins. Ceci a généralement pour conséquence de créer de
nombreuses zones de mémoire libre et occupée, en alternance,
dans la mémoire de votre Amiga. Grâce à ce concept, cet effet
est beaucoup diminué, et surtout la vitesse est notamment
améliorée. Malheureusement ce concept exige l'utilisation du
système 3.0 au moins.
```

1.32 Menu Recherche

```
              Cet ensemble de menus permet de rechercher et de remplacer
des chaînes de caractères dans le texte.


            Rechercher

            Avant

            Après

            Sélection

            Remplacer

            Avant

            Après

            Jusqu'à la fin

            MAJ=min

            Référence

            Aller~à~ligne
```

1.33 Menu Recherche/Rechercher

```
                Cette fonction ne peut fonctionner que si vous avez installé
  la bibliothèque
             bgui
             .library, version 39 ou plus. Celle-ci est
  disponible dans le domaine public (disques DP ou CD-ROM Aminet).

Une fenêtre s'ouvre lors de l'appel de cette fonction : vous y
  trouvez deux boîtes de texte (Chaîne à chercher et chaîne à remplacer)
  ainsi que divers autres gadgets (Avant, Après, Autres, etc...).
La saisie d'une chaîne dans la première de ces boîtes détermine
  naturellement quelle sera la chaîne à chercher.
La recherche s'effectue après la position courante du curseur, en
  sélectionnant le bouton "Après".
Le bouton "Avant" permet la recherche d'une chaîne avant la position
  courante du curseur (en remontant vers le début du texte).
Le bouton "Autre" initialise la boîte de texte.
La boîte "Remplacer", contient la chaîne de remplacement éventuelle,
  enfin le bouton "MAJ=min" permet de déterminer la sensibilité aux
  minuscules et majuscules lors des recherches. Quand il est marqué les mots
  écrits en majuscules (M) ou en minuscules (m) seront indifféremment
  détectés, y compris pour les lettres accentuées. Ainsi la recherche
  permettra par exemple de trouver indifféremment "près", "Près" ou "PRÈS".

Le bouton "Restaurer" permet d'annuler un remplacement effectué après
  avoir cliqué l'un des boutons "Avant" ou "Après" situé sous la boîte
  contenant le texte de remplacement. Cette annulation doit impérativement
  avoir lieu avant que le curseur ne quitte la ligne où a eu lieu ce
  remplacement.

A noter :
    - Une recherche infructueuse est signalée par un clignotement de
l'écran (chaîne non trouvée ou absence de chaîne à chercher).
    - La fenêtre "Recherche" n'a pas forcément a être fermée,  vous
pouvez continuer à travailler normalement dans l'une ou l'autre des
fenêtres textes en y cliquant à l'intérieur.

    Vous pouvez refermer cette fenêtre en cliquant dans le bouton de
fermeture en haut à gauche de la fenêtre ou en tapant sur la touche
Esc. Les raccourcis clavier pour la sélection au clavier d'un des
boutons sont signalés par un caractère souligné (a pour Avant, ...),
d'autre part les touches validant le menu correspondant fonctionnent
de la même façon: n pour "Chercher après" ou b pour "Chercher avant"
par exemple.

Raccourci clavier : AMIGA-F (find) ou CTRL-F pour appeler cette fenêtre
```

1.34 Menu Recherche/(Chercher) Avant

```
                Ce menu permet de lancer la recherche du texte choisi avant la
  position courante du curseur (sans remplacement).
  Si le texte choisi n'a pas été défini, la
             fenêtre de recherche
             ,
  contenant la boîte permettant sa définition est ouverte.
```

Raccourci clavier : AMIGA-B (back), CTRL-B ou F5

1.35 Menu Recherche/(Chercher) Après

Ce menu permet de lancer la recherche du texte choisi après la position courante du curseur (sans remplacement).
Raccourci clavier : AMIGA-N (next), CTRL-N ou F6

1.36 Menu Recherche/Sélection

Le mot ou la phrase sélectionné à l'aide de la souris devient la chaîne à chercher. La recherche s'effectue ensuite vers la fin du texte. Attention, la sélection ne doit se faire que sur une seule ligne à la fois.
Raccourci clavier : AMIGA-J ou CTRL-J

1.37 Menu Recherche/Remplacer

Ces menus permettent de lancer une recherche et un remplacement du texte, s'il a été trouvé.

AVANT : recherche et remplacement avant la position actuelle du curseur.
Raccourci clavier : AMIGA-Z ou CTRL-Z

APRÈS : recherche et remplacement après la position actuelle du curseur.
Raccourci clavier : AMIGA-R (replace) ou CTRL-R

1.38 Menu Recherche/Jusqu'à la fin

Lance la recherche et le remplacement de toutes les occurences du texte choisi, ceci jusqu'à la fin du texte (toujours à partir de la position du curseur) : attention...

Il n'y a pas de raccourci clavier pour cette fonction, qui ne peut être interrompue avant que la recherche n'ait échoué...

1.39 Menu Recherche/MAJ=min

Ce menu, quand il est marqué, permet de lancer une recherche ↩ sans distinguer les majuscules des minuscules dans le texte. Sélectionnez à nouveau ce menu pour annuler son action. À noter que l'état de cette entrée de menu est sauvée dans le fichier de

préférences
s:Configuration.AmiTex

1.40 Menu Recherche/Référence

J'utilise ce menu pour la recherche de documentation quand je fait
de la programmation.
Utilisation : positionnez le curseur sur le DÉBUT du mot pour lequel
vous souhaitez obtenir une aide ou un rappel (par exemple SetAPen)
puis sélectionnez ce menu : le programme cherche alors la présence
du mot SetAPen dans le fichier "s:références_edit", si cette référence
existe le nom du fichier où est située cette documentation ainsi que
son emplacement dans le fichier sont trouvés et une fenêtre est
ouverte avec le texte correspondant.
Si le curseur est placé sur un caractère non alphabétique (ponctuation,
espace ou autre) une demande du terme cherché est faite à l'aide d'une
boîte de requête (utile pour obtenir une aide sur n'importe quel terme
non présent dans le texte).
Vous pouvez également sélectionner le terme à chercher avec la souris,
en faisant glisser le curseur.
Se reporter au chapitre spécifique pour de plus amples explications
sur la structure du fichier "s:références_edit" utilisé pour localiser
les références disponibles.
Raccourci clavier : AMIGA-T ou CTRL-T

1.41 Menu Recherche/Aller à ligne

Ce menu permet de choisir le numéro de la ligne où vous souhaitez
déplacer le curseur. Le numéro de la ligne courante est indiqué dans la
boîte de requête. Pour ma part j'utilise le plus souvent le gadget
proportionnel situé à droite de la fenêtre pour déplacer le curseur dans
le texte (ou même les deux flèches en bas à droite). Cette fonction m'est
utile pour aller à une ligne où est située une erreur signalée par le
compilateur C (celui-ci indiquant son numéro).
Raccourci clavier : AMIGA-G (Go) ou CTRL-G

1.42 Menu Édition

Les premiers menus sont relatifs aux blocs de texte. Avant de
commencer leur description il faut expliquer comment sont sélectionnés
ces blocs. Cette sélection s'effectue en faisant glisser le curseur sur
la zone de texte à sélectionner, tout en maintenant le curseur gauche
de la souris appuyé. Vous pouvez aussi sélectionner un mot en cliquant
deux fois sur la première lettre déterminant le début du mot. En cliquant
sur un caractère non alphabétique, vous sélectionnez la ligne entière.
Pour annuler la sélection d'un bloc pressez la touche ESC ou cliquez
à nouveau n'importe où sur l'écran.

Quand un bloc est déjà sélectionné, vous pouvez l'étendre sans avoir
à faire glisser le curseur tout au long du texte à sélectionner: pressez
une des touches SHIFT avant de cliquer avec le bouton gauche de la souris
à l'emplacement souhaité pour la fin de la sélection. Vous pouvez ainsi
sélectionner tout un ensemble de lignes en positionnant le curseur au
début du texte choisi, vous utilisez ensuite l'ascenseur pour faire défiler
le texte (le bloc reste marqué), quand la fin du bloc désiré apparaît

vous n'avez plus qu'à y cliquer avec le bouton gauche de la souris tout
en maintenant une des touches SHIFT pressée.
De même vous pouvez utiliser la touche CTRL mais c'est alors le
début de la zone sélectionnée qui est déplacé (plus délicat...).

Les fonctions Copier, Coller et Couper utilisent le presse-papiers
de l'Amiga (clipboard.device) rendant ainsi ces opérations possibles
en liaison avec d'autres tâches (traitement de textes ou autres).
Le numéro de l'unité du presse-papiers utilisé peut être changé
à l'aide de la fonction ARexx SETCLIP.
Quand un bloc est sélectionné toute écriture qui s'effectue dans le
bloc lui-même entraîne d'abord l'effacement du bloc.

Deux raccourcis clavier intéressants, en rapport avec les blocs,
sont les suivants :
CTRL HELP permet le déplacement du curseur au début du bloc.
SHIFT HELP permet le déplacement du curseur à la fin du bloc.

Copier

Coller

Couper

Sauver

Indenter~==>

Indenter~<==

==>~MAJUSCULES

==>~minuscules
Effacer~ligne

Restaure~ligne

Insertion

1.43 Menu Édition/Copier

Ce menu permet la recopie du bloc spécifié dans le presse-papiers.
Si l'opération réussit, la sélection est annulée.
Raccourci clavier : AMIGA-C (copier) ou CTRL-C

1.44 Menu Édition/Coller

Le contenu du presse-papiers est copié à l'emplacement actuel
du curseur. Si le curseur est placé dans une zone de texte qui
a été sélectionnée, ce texte est d'abord effacé.

```
  Attention: le
             mode courant
              (insertion ou surimpression) est
  pris en compte, aussi faites attention si vous travaillez en
  mode surimpression, surtout quand vous collez plusieurs lignes
  en même temps...
Raccourci clavier: AMIGA-V ou CTRL-V
```

1.45 Menu Édition/Couper

```
  Ce menu permet d'effacer le bloc spécifié (le bloc est copié dans le
  presse-papiers).
Raccourci clavier : AMIGA-X ou CTRL-X
```

1.46 Menu Édition/Sauver

```
  Ce menu permet de sauvegarder le bloc sélectionné dans un
  fichier (choisi à l'aide d'une requête).
Raccourci clavier : AMIGA-W ou CTRL-W
```

1.47 Menu Édition/Indenter ==>

```
  Ce menu permet l'insertion d'une tabulation sur chacune des
  lignes du bloc (en début de ligne).
Raccourci clavier : AMIGA-TAB (tabulation)
```

1.48 Menu Édition/Indenter <==

```
  Ce menu permet la suppression d'UNE tabulation sur chacune des
  lignes du bloc. Cette suppression a lieu au début de la ligne,
  s'il y existe des espaces.
Raccourci clavier : AMIGA-BACKSPACE (touche de correction)
```

1.49 Menu Édition/==> MAJUSCULES

```
  Ce menu convertit tout le bloc sélectionné en majuscules (lettres
  accentuées comprises, pratique pour insérer des É, ÈÎ...).
  Si aucun texte n'est sélectionné, seul le caractère présent sous
  le curseur est converti.
Raccourci clavier : AMIGA-H (High) ou CTRL-H
```

1.50 Menu Édition/==> minuscules

Ce menu convertit tout le bloc sélectionné en minuscules (lettres accentuées comprises).
Si aucun texte n'est sélectionné, seul le caractère présent sous le curseur est converti.
Raccourci clavier : AMIGA-L (Low) ou CTRL-L

1.51 Menu Édition/Effacer ligne

Ce menu supprime la ligne courante. Cette ligne peut ensuite être "rappelée" en faisant la combinaison ALT-DEL, à l'emplacement où est situé le curseur.
Raccourci clavier : AMIGA-Y ou CTRL-Y ou CTRL-DEL

1.52 Menu Édition/Restaure ligne

Ce menu entraîne l'annulation des modifications apportées à la ligne en cours. Ne sera effectif que si le curseur n'a pas quitté la ligne en question. Cette fonction est utilisable aussitôt après un remplacement (souvent utile).
Raccourci clavier : AMIGA-U (Undo) ou CTRL-U

1.53 Menu Édition/Insertion

Cette marque signale si la frappe s'effectue en mode insertion (par défaut) ou en surimpression. Dans ce mode les lettres entrées au clavier remplacent celles où est situé le curseur. Ce mode est signalé par la couleur du curseur, qui est modifiée dans ce mode (noir au lieu de bleu). Chaque appel de ce menu entraîne sa modification (marqué, non marqué, etc...)
Le mode courant est sauvegardé dans le fichier s:Configuration.amitex lors d'une opération de sauvegarde des

préférences
.
Raccourci clavier : AMIGA-: ou CTRL-:

1.54 Menu Macros

Cette série de menus est dédiée aux
macro-commandes
et
aux
scripts ARexx
.

Mode direct
Saisie, exécution d'une ligne de commande.

Appel~script
Appel d'un script ARexx choisi.

ARexx...
Appel (programmable) de 10 scripts.

1.55 Menu Macros/Mode direct

Ce menu permet la saisie d'une
macro-commande
, puis son exécution. À noter
que vous pouvez également utiliser la combinaison d'une touche ALT et d'une
touche de fonction pour obtenir 10 macros préprogrammées.
Il faut noter que ces séquences programmées sont sauvegardées
dans le fichier "s:Configuration.Amitex", à l'aide du menu

Préférences/Fichier/Sauver
. Celles-ci sont alors rechargées lors de
chaque lancement du programme.

Vous pouvez provoquer l'affichage du résultat de la commande en
faisant commencer le texte spécifiant cette commande par le signe
égal (=).
Exemple: =4*95 affiche le résultat de la multiplication (4 fois 95)
=LINE affiche le numéro de la ligne courante
=COL affiche le numéro de la colonne courante

Raccourci clavier : AMIGA-; ou CTRL-;

1.56 Menu Macros/Appel script

Ce menu permet la saisie d'une commande ARexx (donnez le nom ↩
du
script
sans l'extension ".amitex"). Utilisez la touche SHIFT lors de ↩
la sélection
du menu pour redéfinir la macro.
Raccourci clavier : AMIGA-, ou CTRL-,

1.57 Port ARexx

Le programme ouvre normalement un port de communication avec l'interpréteur
ARexx lors de son lancement, ce port s'appelle AMITEX. Si le programme est
lancé plusieurs fois, le port est appelé AMITEX0 par la seconde tâche, puis
AMITEX1 pour la suivante, et ainsi de suite jusqu'à AMITEX9.

1.58 Menu Macros/ARexx...

```
                Ces 10 menus servent à définir 10 commandes pouvant être ↩
                   lancés par la
  sélection du menu correspondant.
  Lors du premier appel du menu, l'utilisateur définit sa commande.
  Les appels qui suivront permettent alors l'appel du
             script
              défini.
  Le nom du script apparaît ensuite dans le menu. Les 10 définitions peuvent
  être sauvegardées dans le fichier "s:Configuration.AmiTex", et rechargées
  automatiquement lors de l'appel du programme.
Ces menus peuvent être redéfinis en appuyant sur l'une des touches
  SHIFT lors de la sélection du menu.

À noter que l'édition du script défini peut être faite de façon
  automatique dans une nouvelle fenêtre en appelant le menu tout en
  maintenant la touche CTRL appuyée (le nom doit naturellement avoir
  été défini auparavant).

  La commande doit comporter le nom du script à exécuter (toujours sans
  l'extension ".amitex"). Le nom ne peut compter que 11 caractères au maximum.
  Il n'est pas possible de transmettre de paramètres au script.
  Les communications avec ARexx se font grâce au
             port
              AMITEX (ou
  AMITEX0, AMITEX1...AMITEX9 si plusieurs tâches sont exécutées en
  même temps). Utilisez la commande port=ADDRESS() pour connaître le nom
  du port, si nécessaire (voir le script conjugue.amitex pour exemple).
Raccourcis clavier : AMIGA-0 à AMIGA-9
```

1.59 Menu Préférences

```
                Cette série de menus permet de sauvegarder certains réglages ↩
                   du
  programme dans le fichier configuration.amitex. Ce fichier est
  sauvé dans le répertoire assigné "s:".
  Ce fichier est relu automatiquement à chaque démarrage du
  programme, afin de repositionner les réglages comme voulu par
  l'utilisateur.


             Sauver~tabulation

             Copie~texte~sauvé

             Sauver~avec~icône

             Tabulations~=~4

             Choix fonte

             Fichier
```

1.60 Menu Préférences/Sauver tabulation

Cette bascule permet ou interdit l'écriture du caractère TAB (0x9)
dans le fichier lors de la sauvegarde. Si ce menu n'est pas marqué
les tabulations seront remplacées par des espaces. Son utilisation
permet de diminuer de façon conséquente la taille des fichiers.

1.61 Menu Préférences/Copie texte sauvé

Quand ce menu est marqué, lors de la sauvegarde, l'ancienne version
est sauvegardée avec l'extension ".bis" à la fin du nom.

1.62 Menu Préférences/Sauver icône

Si ce menu est marqué une icône est créée lors de la sauvegarde
du texte. Le nom du programme figure dans le champ TOOL-TYPES.
L'icône utilisée est l'icône "Projet" par défaut, stockée dans
le répertoire ENV:.

1.63 Menu Préférences/Tabulations =

Ce menu permet de déterminer l'espace entre les tabulations pour
la touche TAB (4 par défaut) lors de la saisie clavier (valide aussi
pour les menus INDENTER vus précédemment).

1.64 Menu Préférences/Choix fonte

Ce menu permet de choisir la fonte utilisée pour écrire le texte
dans les fenêtres. Chacune des différentes fenêtres peut utiliser une
fonte différente. Cependant cette fonte n'est pas utilisée pour
l'affichage des menus. Seules les fontes non proportionnelles sont admises.
En l'absence de fichier de configuration (voir ci-dessous), c'est la fonte
choisie pour l'écran qui est utilisée. Attention alors à ne pas utiliser
une fonte proportionnelle, le programme ne les gère pas correctement.
Raccourcis clavier : AMIGA-[

1.65 Menu Préférences/Fichier

Ce menu comporte trois rubriques, permettant de sauver ou de
lire les macro-commandes, les noms des
scripts ARexx
choisis, ainsi que
les divers modes (tabulations, sauver avec ou sans icône, sauver avec les
tabulations, copie texte...).

CHARGER : permet de relire le fichier de configuration de votre choix, préalablement sauvé grâce à l'option SAUVER ou SAUVER EN.

SAUVER : permet la sauvegarde des préférences dans le fichier s:Configuration.AmiTex. Ce fichier est relu à chaque chargement du programme.

SAUVER EN : permet la sauvegarde des préférences dans un fichier, de votre choix (requête asl.library).

Lors d'une sauvegarde des préférences, c'est la fonte alors utilisée dans la fenêtre active qui est sauvée (nom et taille).

1.66 Écriture d'une ligne de commande(s)

Les macro-commandes peuvent être exécutées en mode local ou bien par l'intermédiaire d'un script ARexx. Vous pouvez vous reporter aux scripts donnés en exemple, dans le répertoire "s:ARexx". Ceux-ci doivent posséder l'extension ".amitex" à leur nom.
Chacune de ces macro-commandes peut appeler l'une des fonctions ARexx, ou même plusieurs. Certaines d'entre elles nécessitent un ou plusieurs arguments, enfin la plupart d'entre elles retournent un résultat. Le programme permet aussi de manipuler des variables (des types numérique ou chaîne de caractères). De plus vous pouvez définir vos propres fonctions.
L'appel d'une fonction se fait en donnant son nom, suivi d'un ou plusieurs arguments, entourés de parenthèses, éventuellement séparés par des virgules.

Les opérateurs mathématiques classiques sont bien sûr disponibles :

opérateur	signe	priorité
élévation à une puissance	^	10
division	/	9
multiplication	*	9
modulo	%	9
addition	+	8
soustraction	-	8
ET logique	&	6
OU exclusif	?	5
OU logique	|	4
affectation	=	2

Ce dernier opérateur (=) peut être utilisé pour affecter les variables.
Ex : A=2, affecte la valeur 2 à la variable A.
Il est à noter que l'écriture A=B=3 n'est pas admise, elle entraînera un message d'erreur "Affectation impossible".

Autres opérateurs :

```
décalage à gauche : <<
décalage à droite : >>
Ces deux opérateurs ont une priorité égale à 7.
test supérieur : >      test supérieur ou égal : >=
test inférieur : <      test inférieur ou égal : <=
test si différent : <>      test égalité : ==
Les tests renvoient la valeur 1 s'ils sont vérifiés, 0 dans les
  autres cas. Leur priorité est égale à 3.

  Il existe un opérateur particulier qui en fait n'en est pas un, c'est le
  signe :, celui-ci permet en fait de séparer deux formules (ou davantage)
  afin de pouvoir saisir plusieurs affectations de
              variables
               dans une même
  fonction. Ainsi vous pouvez saisir A=2:B=3:C=0 dans une seule cellule au
  lieu d'en utiliser trois. Le résultat renvoyé est alors celui de la
  dernière opération réalisée (0 dans cet exemple). Cette possibilité peut
  également présenter un intérêt pour une fonction FOR (voir plus loin),
  afin de réaliser de multiples initialisations au début d'une boucle.

  Enfin il est à noter que vous pouvez placer un commentaire dans une macro
  comportant une expression mathématique en utilisant l'apostrophe vue
  plus haut.
Exemple:
    GOTO(1,BLOCK(1)) ' Déplacement au début du bloc

Enfin, vous pouvez placer plusieurs appels de fonctions sur une
  même ligne, en les séparant par deux points.
Exemple:
    GOTO(1,1) : WRITE("Essai")
```

1.67 Les variables

```
                Les variables manipulées par AmiTex peuvent être de deux types:
numérique ou chaîne de caractères. Le type est choisi lors de
l'affectation, il ne peut ensuite être changé, à moins de
réinitialiser cette variable (voir fonction
              INIT
              ).

Pour affecter une valeur à une variable il suffit de faire suivre
son nom du signe = et de la valeur à lui affecter.

Exemples:
    A = 1
    B2 = "chaîne de texte"
    A2 = "Première "+B2     donne "Première chaîne de texte"

Les chaînes de caractères doivent être encadrées par des guillemets
s'il s'agit de constantes. Les
              valeurs numériques
               sont limitées aux
entiers, utilisez ARexx pour manipuler des nombres réels.

Les noms de variables peuvent comprendre de 1 à 21 caractères, ils
```

doivent débuter par une lettre, les caractères suivants pouvant être
des lettres (accentuées ou non), des nombres ou le caractère _.

Exemples de noms valides:
    ESSAI
    TYPE_DONNÉE
    LIGNE1

À noter que le programme convertit normalement les noms de fonctions
et de variables en majuscules lors de la saisie, aussi n'y aura-t-il
pas de différence entre deux noms comme variable et VARIABLE (ou même
Variable), cependant dans un
              script ARexx
              , utilisez de préférence les
majuscules car les chaînes issues de l'interpréteur ne sont pas
forcément converties, et le programme différenciera alors les noms
écrits en minuscules et/ou en majuscules.

1.68 Les variables numériques

Les nombres manipulés par Amitex sont limités aux nombres entiers.
Ceux ci sont des entiers longs signés, sur 32 bits. La valeur
maximale est doc de 2^31-1 (2147483647) et la valeur minimale
-2^31 (-2147483648). Utilisez les capacités de l'interpréteur
ARexx pour manipuler des nombres en virgule flottante.

1.69 Les chaînes de caractères

Les chaînes de caractères comprennent un nombre quelconque de
caractères (théoriquement limité uniquement par la capacité
mémoire de votre micro-ordinateur), en commençant par les
chaînes nulles qui ne comprennent aucun caractère. Pour
définir une chaîne de caractères, vous devez encadrer son
contenu par deux guillemets. Si cette chaîne comprend
elle-même un ou plusieurs guillemets, ceux-ci doivent être
doublés.

Exemples:
    "Ceci est une chaîne de caractères"
    ""
    "C'est un guillemet "" !"

1.70 Structure des scripts ARexx

Les scripts ARexx sont des fichiers ASCII, se conformant à un
format permettant leur interprétation par ce logiciel.
Ils peuvent inclure toutes les fonctions spécifiques à ARexx
et à ses bibliothèques (reportez-vous à leurs documentations
pour plus de précisions), ainsi que toutes les commandes
supportées par le programme AmiTex (une centaine, plus les

fonctions éventuellement définies par vos soins).

Emplacement
Ces scripts doivent se situer soit dans le répertoire assigné REXX: (conseillé), soit dans le répertoire courant (normalement le même que celui du programme AmiTex). Il est conseillé de les nommer avec l'extension .AmiTex, mais ce n'est pas obligatoire.

Format
Ces scripts doivent toujours commencer par une remarque incluse entre les caractères /* et */, comme en langage C.

Les commandes AmiTex doivent impérativement figurer en majuscules. Ce sont les mêmes que celles qui sont utilisées en
mode local
, cependant leur analyse par ARexx nécessite quelques précautions, ainsi ces commandes doivent impérativement être encadrées par des apostrophes (') ou des guillemets, afin de les différencier des fonctions internes à ARexx.

Exemple
La structure suivante est recommandée pour ces scripts:

```
/* Ce fichier vous donne un exemple de structure possible
pour un fichier de macros appelé par AmiTex, par le menu Macros */

options results     /* indispensable pour récupérer le résultat des macros */

signal on error     /* pour l'interception des erreurs */
signal on syntax

/* votre programme doit être situé dans cette zone */

exit

/* Traitement des erreurs, interruption du programme */
syntax:
erreur=RC
'MESSAGE("Erreur de syntaxe"+CHR(10)+"en ligne 'SIGL'"+CHR(10)+"'errortext(erreur) ↩
   '")'
exit

error:
'MESSAGE("Erreur en ligne 'SIGL'")'
exit
```

Cet exemple existe déjà prêt dans le répertoire Rexx, sous le nom squelette.amitex.

1.71 Liste des fonctions ARexx

ABS

```
 valeur absolue d'un nombre

ASC
 code ASCII d'un caractère

ASK
 saisie d'une chaîne de caractères au clavier

BEGLINE
 déplacement du curseur sur le premier caractère de la ligne

BLOCK
 renvoie le numéro de colonne ou de ligne du bloc marqué

CALL
 appel d'un script ARexx

CATLINES
 regroupement de deux lignes de texte

CHR
 renvoie le caractère possédant le code ASCII spécifié

CLIPUNIT
 choix de l'unité du presse-papiers (clipboard.device)

CLOSE
 ferme la fenêtre texte spécifiée

COL
 numéro de colonne où est situé le curseur

COPY
 copie d'un texte dans une unité du presse-papiers

CUTLINE
 coupure de la ligne de texte

DATE
 renvoie la date actuelle

DAYS
 comparaison de deux dates

DEF
 définition d'une nouvelle fonction

DELBEGIN
 effacement du début de la ligne

DELCHARS
 effacement de caractères

DELEND
 effacement de la fin de la ligne

DELLEFT
```

```
 effacement des mots à gauche du curseur

DELPREV
 effacement des caractères précédant le curseur

DELRIGHT
 effacement du mot à droite du curseur

ENDLINE
 déplacement du curseur à la fin de la ligne

EXEC
 interprétation d'une chaîne de caractères

FILENAME
 nom du fichier complet

FILEPART
 nom du fichier

FIND
 recherche de texte

FONTNAME
 renvoie le nom de la fonte de caractères utilisée

FONTSIZE
 renvoie la taille de la fonte de caractères utilisée

FOR
 traitement d'une boucle

GOTO
 déplacement du curseur

HELP
 affichage d'une aide AmigaGuide

IF
 test

INIT
 initialisation de variables

INSLINES
 insertion de lignes vides

INSTEXT
 insertion du texte spécifié

LEN
 longueur d'une chaîne de caractères

LINE
 numéro de ligne où est situé le curseur

LOAD
```

```
 chargement d'un texte

LOADPREF
 chargement d'un fichier de préférences

LOADREF
 chargement d'une référence

LOCK
 verrouillage saisies utilisateur

LOWER
 conversion d'un texte en minuscules

MACRO
 appel d'une séquence programmée

MARK
 marquage d'un bloc de texte

MENU
 exécution d'un menu

MESSAGE
 affichage d'un message

MESURE
  renvoie une dimension de la fenêtre

MODEDIT
 fixe le mode insertion/surimpression

MODIF
 test modification texte

NBLINES
 nombre de lignes du texte

NBTEXT
 nombre de textes présents en mémoire

NEW
 ouverture d'une nouvelle fenêtre

OPEN
 ouverture de texte(s)

PASTE
 collage du contenu d'une unité du presse-papiers

PICKCOL
 sélection d'une colonne de texte à l'aide de la souris

PICKLINE
 sélection d'une ligne de texte à l'aide de la souris

POS
```

```
 position d'une chaîne de caractères

PRINT
 impression texte

READCHAR
 lecture des caractères courants

READLINE
 lecture de la ligne spécifiée

REPLACE
 remplacement d'un texte

REQFILE
 choix d'un fichier

REQTEXT
 choix d'un texte

REQUEST
 affichage d'un message, choix OUI/NON

RESET
 réinitialisation de variables

SAVE
 sauvegarde d'un texte

SAVBLOCK
 sauvegarde du bloc de texte marqué

SAVECOPY
 test/choix sauvegarde copie fichier texte

SAVEICON
 test/choix sauvegarde avec création icône

SAVEPREF
 sauvegarde fichier préférences

SAVETABS
 test/choix sauvegarde tabulations

SEARCH
 recherche de texte avec motifs #?[]()

SECURITY
 détermination d'un nombre de boucles maximum avant débordement

SELECT
 choix d'une option parmi plusieurs

SELFILE
 sélection du texte

SELTEXT
```

```
 sélection d'un texte

SETFIND
 choix de la chaîne à rechercher

SETREP
 choix de la chaîne de remplacement

SETFONT
 choix d'une fonte de caractères

SETTABS
 nombre de caractères associés à une tabulation

SGN
 test du signe d'un nombre

SORT
 tri de lignes de texte

STR
 conversion d'un nombre en chaîne de caractères (décimal)

STRBIN
 conversion d'un nombre en chaîne de caractères (binaire)

STRHEX
 conversion d'un nombre en chaîne de caractères (hexadécimal)

SUBS
 renvoie un sous-ensemble d'une chaîne de caractères

SUPBLOCK
 effacement du bloc de texte

SUPLINES
 suppression de lignes de texte

TESTCASE
 mode de recherche (sensibilité aux majuscules/minuscules)

TEXTMARK
 renvoie le texte marqué

TIME
 renvoie l'heure courante

TITLE
 choix du titre affiché dans la fenêtre

TOBACK
 renvoie la fenêtre active en arrière-plan

TOFRONT
 renvoie la fenêtre active en avant-plan

TRACE
```

```
 marquage d'un emplacement

TYPE
 renvoie le type d'un caractère

UNLOCK
 annule le verrouillage des actions utilisateur

UNMARK
 annulation du marquage d'un texte

UPPER
 conversion d'un texte en majuscules

VAL
 conversion d'une chaîne de caractères en nombre

VERSION
 numéro de version du programme

WHILE
 boucle tant_que...

WINDOW
 dimensionne la fenêtre texte

WLEFT
 déplacement d'un mot vers la gauche

WORD
 lecture du mot courant

WRIGHT
 déplacement d'un mot vers la droite

WRITE
 écriture d'un texte

WRITETAB
 écriture tabulations

Index thématique
```

1.72 Classement thématique des fonctions ARexx

```
 Traitement des chaînes de caractères


 Édition, modification du texte


 Fonctions portant sur des blocs de texte
```

Gestion, déplacement du curseur

Recherche/Remplacement

Fonctions mathématiques

Fonctions interactives

Gestion des fenêtres

Gestion des préférences

Fonctions diverses

1.73 Fonctions ARexx de traitement des chaînes de caractères

ASC
code ASCII d'un caractère

CHR
renvoie le caractère possédant le code ASCII spécifié

LEN
longueur d'une chaîne de caractères

LOWER
conversion d'un texte en minuscules

POS
position d'une chaîne de caractères

STR
conversion d'un nombre en chaîne de caractères (décimal)

STRBIN
conversion d'un nombre en chaîne de caractères (binaire)

STRHEX
conversion d'un nombre en chaîne de caractères (hexadécimal)

SUBS
renvoie un sous-ensemble d'une chaîne de caractères

TYPE
renvoie le type d'un caractère

UPPER

```
 conversion d'un texte en majuscules

VAL
 conversion d'une chaîne de caractères en nombre
```

1.74 Fonctions ARexx permettant l'édition du texte

```
CATLINES
 regroupement de deux lignes de texte

CUTLINE
 coupure de la ligne de texte

DELBEGIN
 effacement du début de la ligne

DELCHARS
 effacement de caractères

DELEND
 effacement de la fin de la ligne

DELLEFT
 effacement des mots à gauche du curseur

DELPREV
 effacement des caractères précédant le curseur

DELRIGHT
 effacement du mot à droite du curseur

INSLINES
 insertion de lignes vides

MODEDIT
 fixe le mode insertion/surimpression

NBLINES
 nombre de lignes du texte

READCHAR
 lecture des caractères courants

READLINE
 lecture de la ligne spécifiée

SETTABS
 nombre de caractères associés à une tabulation

SORT
 tri de lignes de texte

SUPLINES
 suppression de lignes de texte
```

```
WORD
 lecture du mot courant

WRITE
 écriture d'un texte

WRITETAB
 écriture tabulations
```

1.75 Fonctions ARexx permettant de gérer les blocs de texte

```
BLOCK
 renvoie le numéro de colonne ou de ligne du bloc marqué

CLIPUNIT
 choix de l'unité du presse-papiers (clipboard.device)

COPY
 copie d'un texte dans une unité du presse-papiers

MARK
 marquage d'un bloc de texte

PASTE
 collage du contenu d'une unité du presse-papiers

SAVBLOCK
 sauvegarde du bloc de texte marqué

SUPBLOCK
 effacement du bloc de texte

TEXTMARK
 renvoie le texte marqué

UNMARK
 annulation du marquage d'un texte
```

1.76 Fonctions ARexx permettant de gérer les déplacements du curseur

```
BEGLINE
 déplacement du curseur sur le premier caractère de la ligne

ENDLINE
 déplacement du curseur en fin de ligne

COL
 numéro de colonne où est situé le curseur
```

```
GOTO
 déplacement du curseur

LINE
 numéro de ligne où est situé le curseur

NBLINES
 nombre de lignes du texte

TRACE
 marquage d'un emplacement

WLEFT
 déplacement d'un mot vers la gauche

WRIGHT
 déplacement d'un mot vers la droite
```

1.77 Fonctions ARexx de gestion de la recherche et du remplacement de texte

```
FIND
 recherche de texte

REPLACE
 remplacement d'un texte

SEARCH
 recherche de texte avec motifs #?[]()

SETFIND
 choix de la chaîne à rechercher

SETREP
 choix de la chaîne de remplacement

TESTCASE
 mode de recherche (sensibilité aux majuscules/minuscules)
```

1.78 Fonctions ARexx de gestion des préférences

```
FONTNAME
 nom de la fonte de caractères utilisée

FONTSIZE
 taille de la fonte de caractères utilisée

LOADPREF
 chargement d'un fichier de préférences

SAVECOPY
```

```
 test/choix sauvegarde copie fichier texte

SAVEICON
 test/choix sauvegarde avec création icône

SAVEPREF
 sauvegarde fichier préférences

SAVETABS
 test/choix sauvegarde tabulations

SETTABS
 nombre de caractères associés à une tabulation

SETFONT
 choix d'une fonte de caractères
```

1.79 Fonctions ARexx de calcul

```
ABS
 valeur absolue d'un nombre

FOR
 traitement d'une boucle

IF
 test

INIT
 initialisation de variables

RESET
 réinitialisation de variables

SECURITY
 nombre de boucles maximum avant débordement

SGN
 test du signe d'un nombre

STR
 conversion d'un nombre en chaîne de caractères (décimal)

VAL
 conversion d'une chaîne de caractères en nombre

WHILE
 boucle tant_que...
```

1.80 Fonctions ARexx permettant le dialogue avec l'utilisateur

```
ASK
 saisie d'une chaîne de caractères au clavier

LOCK
 verrouillage saisies utilisateur

MESSAGE
 affichage d'un message

PICKCOL
 sélection d'une colonne de texte à l'aide de la souris

PICKLINE
 sélection d'une ligne de texte à l'aide de la souris

REQFILE
 choix d'un fichier

REQUEST
 affichage d'un message, choix OUI/NON

SELECT
 choix d'une option parmi plusieurs

UNLOCK
 annule le verrouillage des actions utilisateur
```

1.81 Fonctions ARexx diverses

```
CALL
 appel d'un script ARexx

DATE
 renvoie la date actuelle

DAYS
 comparaison de deux dates

EXEC
 interprétation d'une chaîne de caractères

HELP
 affichage d'une aide AmigaGuide

MACRO
 appel d'une séquence programmée

MENU
 exécution d'un menu

TIME
 renvoie l'heure courante
```

```
VERSION
 numéro de version du programme
```

1.82 Fonctions ARexx de gestion des fenêtres

```
CLOSE
 ferme la fenêtre texte spécifiée

FILENAME
 nom du fichier complet

FILEPART
 nom du fichier

FONTNAME
 nom de la fonte de caractères utilisée

FONTSIZE
 taille de la fonte de caractères utilisée

INSTEXT
 insertion du texte spécifié

LOAD
 chargement d'un texte

MESURE
  renvoie une dimension de la fenêtre

MODIF
 test modification texte

NBTEXT
 nombre de textes présents en mémoire

NEW
 ouverture d'une nouvelle fenêtre

OPEN
 ouverture de texte(s)

PRINT
 impression texte

REQTEXT
 choix d'un texte

SAVE
 sauvegarde d'un texte

SELFILE
 sélection du texte
```

```
SELTEXT
 sélection d'un texte

SETFONT
 choix d'une fonte de caractères

TITLE
 choix du titre affiché dans la fenêtre

TOBACK
 renvoie la fenêtre active en arrière-plan

TOFRONT
 renvoie la fenêtre active en avant-plan

WINDOW
 dimensionne la fenêtre texte
```

1.83 Fonction ARexx DEF

```
Vous pouvez définir un nombre quelconque de fonctions utilisant
  des fonctions internes ainsi que n'importe quel opérateur.
  Une de ces fonctions peut même faire appel à une autre fonction
  précédemment définie. Les arguments peuvent être d'un type quelconque,
  il doivent simplement correspondre aux types attendus par les fonctions
  qui seront appelées ou être compatibles avec les opérateurs utilisés.

Il ne peut y avoir plus d'une déclaration dans une même ligne.
La forme de ces définitions est de la forme suivante:
    DEF nom_fonction(argument,...) = définition des opérations.
Le mot clé DEF doit impérativement commencer la définition, sans
  aucun espace préalable.

Le nom des fonctions peut comprendre de 1 à 13 caractères
  alphanumériques, y compris les lettres accentuées et le soulignement (_).
  Le premier caractère doit cependant toujours être une lettre.
Les fonctions internes ne peuvent être redéfinies.
  Le nombre des arguments est limité à 15 au maximum.
Ce nombre est fixe pour une définition donnée (vous ne pouvez pas
  déclarer de fonction possédant un nombre variable d'arguments).
Il doit y avoir au moins un argument dans la définition, mais celui-ci
  peut ne pas être utilisé.
Les arguments doivent bien entendu posséder des noms distincts.

  Exemples:
Création d'un patronyme complet
    DEF PATRONYME(NOM,PRENOM) = UPPER(NOM)+" "+PRENOM

La redéfinition d'une fonction ayant déjà été définie est possible,
  la nouvelle définition remplace alors l'ancienne. Ceci peut être utile
  lorsque vous essayez de définir une fonction complexe.
```

1.84 Fonction ARexx ABS

```
                  ABS(nombre)
Cette fonction renvoie la valeur absolue du nombre donné en
argument. Le nombre peut appartenir à une
              variable
              .

Exemples:
    ABS(-8)     renvoie 8
    ABS(4)      renvoie 4
    ABS(N)      renvoie la valeur absolue du nombre N

Rappel: les nombres doivent être du type entier.
```

1.85 Fonction ARexx ASC

```
                  ASC(texte,rang)
Cette fonction renvoie le code ASCII du caractère appartenant
à la chaîne de texte passée en argument, du rang spécifié.
Le rang doit être compris entre 1 et la longueur de la chaîne
passée en argument.

Exemples:
    ASC("essai",1)  renvoie 101 (code ASCII du caractère e)
    ASC(TEXTE,LEN(TEXTE))   renvoie le code ASCII du dernier
          caractère de la
              variable
               TEXTE.

Voir aussi:
              CHR
```

1.86 Fonction ARexx ASK

```
  ASK(texte)
Ouvre une requête contenant le texte spécifié. La boîte de requête
contient une boîte de texte dans laquelle l'utilisateur peut saisir
une chaîne de caractères. Le texte saisi est renvoyé comme résultat
si l'utilisateur clique sur le bouton Valide, sinon une chaîne nulle
est renvoyée (clic sur le bouton Annule).
Le texte affiché en titre peut contenir de une à treize lignes
séparées par des sauts de lignes.

Exemple:
    ASK("Entrez le premier"+CHR(10)+"mot puis frappez"+CHR(10)+"sur ENTREE")
```

1.87 Fonction ARexx BEGLINE

```
                  BEGLINE(ligne)
Déplace le curseur sur le premier caractère de la ligne spécifiée.
À noter que le numéro de la première ligne est le 1. La ligne
courante, où est située le curseur, est désignée par la fonction

              LINE
              , le nombre total de lignes du texte courant est donné par la
fonction
              NBLINES
              .
Cette fonction renvoie le numéro de colonne où se place le curseur.

Exemple:
    BEGLINE(LINE)   déplacement sur le premier caractère de la ligne courante
    BEGLINE(10)     déplacement sur le premier caractère de la dixième ligne
```

1.88 Fonction ARexx BLOCK

```
  BLOCK(indice)
Cette fonction sert à déterminer les coordonnées du début et de
la fin de la zone de texte marquée (sélectionnée).

La valeur renvoyée est nulle si aucun texte n'est sélectionné,
sinon la valeur renvoyée dépend de l'argument:
- elle correspond au numéro de colonne de début si l'argument vaut 0,
- elle correspond au numéro de ligne de début si l'argument vaut 1,
- elle correspond au numéro de colonne de fin si l'argument vaut 2,
- elle correspond au numéro de ligne de fin si l'argument vaut 3.
Toute autre valeur de l'argument entraîne un message d'erreur.

Exemple:
    'BLOCK(0)'
    if result=0 then do
  'MESSAGE("Sélectionnez une zone"+CHR(10)+"de texte !")'
  exit
    end
```

1.89 Fonction ARexx CALL

```
                  CALL(script, argument1, argument2...)
Appel d'un script ARexx. Le nom du script, donné en argument, n'a pas
à inclure l'extension ".amitex". Cette fonction n'est pas utile dans
un script ARexx mais peut servir dans une
              macro-commande
                (ALT-Fx ou
macro locale).
L'appel peut être fait en spécifiant de 0 à 15 arguments (
              numériques
                ou chaînes de caractères).

L'exécution du script se fait en asynchrone, c'est à dire que la main
```

```
est rendue au programme appelant avant la fin de l'exécution du script.
La valeur renvoyée est le nom du script (sans grand intérêt)...

Exemples:
    'CALL("editscript")'
    'CALL("multiplier","1.5","15.25")
```

1.90 Fonction ARexx CATLINES

```
  CATLINES(ligne)
Cette fonction permet de regrouper (concaténer) la ligne spécifiée
et la ligne qui la suit. Le résultat de la fonction est égal à 1
si tout s'est bien passé, 0 sinon.

À noter que le numéro de la première ligne est le 1. La ligne
courante, où est située le curseur, est désignée par la variable
LINE, le nombre total de lignes du texte courant est donné par la
variable NBLINES.

Exemples:
    CATLINES(1)     regroupe les première et seconde lignes
    CATLINES(LINE)  regroupe la ligne courante et la suivante
```

1.91 Fonction ARexx CHR

```
                  CHR(code)
Renvoie le caractère possédant le code ASCII spécifié.

Utilisé pour obtenir un saut de ligne (CHR(10)) ou un caractère
spécial non présent sur le clavier. Le code peut varier de 1 à
255 maxi.

Voir aussi:
              ASC
```

1.92 Fonction ARexx CLIPUNIT

```
  CLIPUNIT(unité)
Cette fonction permet de choisir l'unité du presse-papiers
utilisé pour les opérations de couper/coller. L'unité de
presse-papiers peut varier de 0 à 255, soit 256 tampons
utilisables.
L'unité de presse-papiers généralement utilisée est l'unité
0. Elle correspond au clipboard.device, géré par le système.

La valeur renvoyée correspond au numéro de l'unité qui été
utilisée AVANT l'appel à cette fonction. Si vous voulez
connaître celui-ci sans en changer passez une valeur négative
pour l'argument.
```

```
Exemple de script:
    'CLIPUNIT(1)'       choix d'une nouvelle unité (numéro 1)
    clip=RESULT   conserver le numéro de l'ancienne unité
    ...
    'CLIPUNIT('clip')'  remettre l'ancienne unité
```

1.93 Fonction ARexx CLOSE

```
                  CLOSE(fenêtre)
Cette fonction provoque la fermeture de la fenêtre d'indice
spécifié. Chaque fenêtre possède un indice différent, en
commençant par l'indice 0 ( voir
              SELTEXT
              ).
La valeur renvoyée correspond au nombre de fenêtres texte
restant présentes en mémoire.

Utilisez la commande
              MENU
              ("Quitter") pour fermer toutes les
fenêtres et entraîner la fin du programme.

Pour cacher une fenêtre, sans perdre son texte en mémoire,
utilisez la commande MENU("Cacher"), pour la réduire à sa
taille minimale, utilisez la commande MENU("Réduire").
```

1.94 Fonction ARexx COL

```
  COL
Cette fonction permet de connaître le numéro de colonne où est
situé le curseur. Elle n'a besoin d'aucun paramètre.

Exemple:
    GOTO(COL+1,LINE)    ' déplacement d'un caractère vers la droite
```

1.95 Fonction ARexx COPY

```
                  COPY(clip,chaîne)
Cette fonction permet de recopier une chaîne de caractères
dans l'unité du presse-papiers spécifiée. Le numéro de clip
peut varier de 0 à 255. L'unité de clip utilisée pour les
opérations de couper/coller n'est pas modifiée.
Utilisez la fonction
              PASTE
               pour coller le texte ainsi collé
ou choisissez l'unité comme unité par défaut (SETCLIP) puis
utilisez le menu
              Édition/Coller
              .
```

La valeur renvoyée est égale à 0 si tout se passe normalement.

1.96 Fonction ARexx CUTLINE

CUTLINE(x)
Provoque la coupure de la ligne à l'emplacement où est
situé le curseur. Si l'argument est positif, la fin de
la ligne est alignée sur le premier caractère de la ligne
courante sinon elle est placée au début de la ligne.
La valeur renvoyée est positive si tout se passe bien.

1.97 Fonction ARexx DATE

DATE(jour)
Renvoie la date courante. Si l'argument jour est différent
de zéro, le jour de la semaine est inclus.

Exemples:
DATE(0) renvoie 26-Déc-96
DATE(1) renvoie Jeudi 26-Déc-96

Voir aussi:
DAYS
,
TIME
.

1.98 Fonction ARexx DAYS

DAYS(date1,date2)
Renvoie le nombre de jours séparant les deux dates.
Si la première date est antérieure à la seconde la valeur renvoyée
est positive, sinon elle est négative (ou nulle si les deux dates
concordent). Le format de la date doit correspondre à celui
obtenu par l'instruction
DATE(0)
, c'est à dire que le jour de
la semaine ne doit pas être inclus. Cependant vous pouvez
spécifier un jour de la semaine, mais alors sans la date,
ainsi DAYS ("Mercredi", "18-Sep-95") renvoie le nombre de
jours séparant le dernier mercredi passé du 18 septembre 95.
Vous pouvez également spécifier "hier" ou "aujourd'hui".
À noter que les dates doivent être postérieures au 1er janvier 78.
Ainsi la date suivante 20-oct-14, correspond au 20 octobre de
l'année 2014.

Voir aussi :
TIME
.

1.99 Fonction ARexx DELBEGIN

```
  DELBEGIN(0)
Efface le début de la ligne (de la colonne 1 à la colonne
où se situe le curseur). La valeur de l'argument n'a aucune
importance.
Renvoie le numéro de la colonne où est situé le curseur à
la fin de l'opération (normalement 1).
```

1.100 Fonction ARexx DELCHARS

```
  DELCHARS(nombre_caractères)
Effacement du nombre de caractères spécifié, à partir de
l'emplacement du curseur. Si la fin de la ligne ne comprend
plus de caractères, la ligne est jointe avec la ligne qui
la suit.
Renvoie le numéro de la colonne où est situé le curseur à
la fin de l'opération (normalement inchangé).
```

1.101 Fonction ARexx DELEND

```
  DELEND(0)
Effacement de la fin de la ligne, à partir de
l'emplacement du curseur.
Renvoie le numéro de la colonne où est situé le curseur à
la fin de l'opération (normalement inchangé).
```

1.102 Fonction ARexx DELLEFT

```
  DELLEFT(nombre_mots)
Effacement du nombre de mots situés à gauche du curseur.
Attention, les caractères ne faisant pas partie d'un mot,
comme les espaces, les signes de ponctuation, etc, sont
comptés comme des mots. Si le curseur atteint la première
colonne l'opération est interrompue.
Renvoie le numéro de la colonne où est situé le curseur à
la fin de l'opération.
```

1.103 Fonction ARexx DELPREV

```
  DELPREV(nombre_caractères)
Effacement du nombre de caractères spécifiés, situés avant
le curseur. Si le curseur atteint la première colonne, le
curseur se déplace à la fin de la colonne précédente, et
l'opération se poursuit.
Renvoie le numéro de la colonne où est situé le curseur à
la fin de l'opération.
```

1.104 Fonction ARexx DELRIGHT

```
  DELRIGHT(nombre_mots)
Effacement du nombre de mots situés à droite du curseur.
Attention, les caractères ne faisant pas partie d'un mot,
comme les espaces, les signes de ponctuation, etc, sont
comptés comme des mots. Si le curseur atteint la dernière
colonne l'opération est interrompue.
Renvoie le numéro de la colonne où est situé le curseur à
la fin de l'opération.
```

1.105 Fonction ARexx ENDLINE

```
                  ENDLINE(ligne)
Déplacement du curseur à la fin de la ligne spécifiée.
À noter que le numéro de la première ligne est le 1. La ligne
courante, où est située le curseur, est désignée par la fonction

              LINE
              , le nombre total de lignes du texte courant est donné par la
fonction
              NBLINES
              .
Renvoie le numéro de la colonne où est situé le curseur à
la fin de l'opération.

Exemple:
    ENDLINE(LINE)       déplacement à la fin de la ligne courante
    ENDLINE(NBLINES)    déplacement à la fin du texte
```

1.106 Fonction ARexx EXEC

```
  EXEC(chaîne de caractères)
Demande l'interprétation et l'exécution de la chaîne de
caractères passée en argument, comme s'il s'agissait
d'une ligne de commande.
Le résultat dépend naturellement du contenu de la
chaîne passée en argument.

Exemple:
    EXEC(READLINE(LINE))    interprète la ligne courante
```

1.107 Fonction ARexx FILENAME

```
  FILENAME(nom)
Renomme la fenêtre courante, avec le nom spécifié.
Si une chaîne nulle ("") est passée en argument, cette
fonction renvoie le nom de la fenêtre, y compris le
chemin complet.
```

1.108 Fonction ARexx FILEPART

```
  FILEPART(nom)
Renomme la fenêtre courante, en conservant le chemin.
Si une chaîne nulle ("") est passée en argument, cette
fonction renvoie le nom de la fenêtre, sans le
chemin complet.

Exemple:
    Supposons que le fichier courant soit "RAM:sources/x.c"
    la fonction FILEPART("") renverrait x.c, alors que la
    fonction FILEPART("nn.c") renommerait le fichier
    "RAM:sources/nn.c".
```

1.109 Fonction ARexx FIND

```
                  FIND(occurences)
Lance la recherche du texte (spécifié par SETFIND ou par
le menu
              Rechercher
              ).
Si l'argument occurences est positif, la recherche s'effectue
après l'emplacement du curseur, alors que si cet argument est
négatif, la recherche s'effectue vers le début du texte.
Dans les deux cas la recherche s'arrête quand le nombre
d'occurences spécifié a été atteint.

La valeur renvoyée est égale au nombre de fois où la recherche
a abouti.
```

1.110 Fonction ARexx FONTNAME

```
                  FONTNAME(x)
Renvoie le nom de la fonte utilisée dans la fenêtre courante.
L'argument peut prendre n'importe quelle valeur. Utilisez de
préférence l'argument
              SELTEXT
              (-1), en vue d'une compatibilité
ultérieure.

Exemple:
    FONTNAME(SELTEXT(-1))   renvoie topaz.font (par exemple)
```

1.111 Fonction ARexx FONTSIZE

```
                  FONTSIZE(x)
Renvoie la taille de la fonte utilisée dans la fenêtre courante.
L'argument peut prendre n'importe quelle valeur. Utilisez de
préférence l'argument
```

```
              SELTEXT
              (-1), en vue d'une compatibilité
ultérieure.

Exemple:
    FONTNAME(SELTEXT(-1))   renvoie 11 (par exemple)
```

1.112 Fonction ARexx FOR

```
                  FOR(init,condition_fin,action1,...) cette fonction permet de ↩
                     définir des
boucles. Le premier argument (init) est exécuté une seule fois,
quand l'appel de la fonction vient d'être fait. Le second argument
définit la condition de fin de boucle. Enfin le troisième argument
ainsi que les arguments suivants s'ils existent sont évalués à
chaque exécution de la boucle.

Exemples:
    FOR (I=0, I<10, I=I+1)
    Dans cet exemple la
              variable
               I est initialisée à la valeur 0,
    tant que cette valeur est inférieure à 10, on incrémente cette
    valeur. I va donc prendre successivement les valeurs 1 à 10.

    FOR (I=0, I<10, I=I+1, WRITE(STR(I)))
    Cette formule comprend une instruction supplémentaire
    permettant d'écrire la valeur de la variable I à l'emplacement
    du curseur.

    N=100:FOR (I=0:J=0, I<=N, J=J+I, I=I+1):J
    Cette formule permet de calculer la somme des 100 premiers
    nombres. Le résultat de la somme est renvoyé (:J à la fin).

Remarque: une boucle bloquée peut être interrompue par deux
moyens: soit par le nombre de boucles maximal (défini par la
fonction
              SECURITY
              , soit en appuyant simultanément sur les trois
touches CTRL, ALT et ESC.
```

1.113 Fonction ARexx GOTO

```
                  GOTO(colonne,ligne)
Déplacement du curseur à l'emplacement spécifié.
    À noter que le numéro de la première ligne est le 1. La ligne
courante, où est située le curseur, est désignée par la fonction

              LINE
              , le nombre total de lignes du texte courant est donné par la
fonction
              NBLINES
```

```
              . La colonne courante est désignée par la fonction

              COL
              .
    Cette fonction renvoie le contenu de la ligne.

Exemples:
    GOTO(COL,LINE-1)    déplacement sur la ligne du dessus
    GOTO(1,NBLINES)     déplacement sur la dernière ligne
    DEF DOWN(LIGNES)=GOTO(COL,LINE+LIGNES)
    DEF UP(LIGNES)=GOTO(COL,LINE-LIGNES)
    DEF RIGHT(COLONNES)=GOTO(COL+COLONNES,LINE)
    DEF LEFT(COLONNES)=GOTO(COL-COLONNES,LINE)
  Les quatre derniers exemples permettent de définir
    des fonctions permettant de déplacer le curseur de
    façon relative à son emplacement actuel.
```

1.114 Fonction ARexx HELP

```
  HELP(node)
Cette fonction permet l'appel d'AmigaGuide, comme par le
menu "Projet/Aide spécifique". L'argument doit être le
nom d'un noeud (node) du fichier AmiTex.guide. Vous
pouvez, en particulier, spécifier n'importe quel nom
de fonction ou de menu.

Exemples:
    HELP("GOTO")
    HELP("Copier")
```

1.115 Fonction ARexx IF

```
  IF(x, a1, a2)
Si x est différent de zéro, renvoie a1 sinon renvoie a2.
Seul a1 OU a2 sera évalué, selon le résultat de x. À noter que les
arguments a1 et a2 peuvent être de n'importe quel type, ils
peuvent également faire appel à d'autres fonctions, y compris
d'autres fonctions IF.

Exemples:
    IF(A>B,A,B)                 renvoie la valeur maximale
    DEF MIN(A,B)=IF(A<B,A,B)    définition fonction MIN
```

1.116 Fonction ARexx INIT

```
                INIT(variable,...)
    Cette fonction est identique à la fonction
              RESET
              ,
cependant le type est réinitialisé, c'est à dire que la
```

```
              variable
               pourra ensuite prendre n'importe quel type autorisé
(numérique ou chaîne de caractères).
    Le nombre d'arguments est quelconque.
```

1.117 Fonction ARexx INSLINES

```
  INSLINES(lignes)
Insertion du nombre de lignes spécifié AVANT la ligne où
est situé le curseur. Renvoie le nombre de lignes ayant
été effectivement insérées.
```

1.118 Fonction ARexx INSTEXT

```
  INSTEXT(nom_fichier)
Insère le texte spécifié avant la ligne où est situé
le curseur.
Renvoie le nombre de lignes ayant été insérées.

Exemple:
    'GOTO(LEN(READLINE(NBLINES))+1,NBLINES)'
    'WRITE(CHR(10))'
    'INSTEXT("Texte")'
    Ce petit script permet d'ajouter un fichier de texte,
    possédant le nom Texte, dans cet exemple, à la fin de
    la fenêtre courante.
```

1.119 Fonction ARexx LEN

```
  LEN(chaîne)
Renvoie la longueur de la chaîne de caractères passée
en argument.
```

1.120 Fonction ARexx LINE

```
  LINE
Cette fonction renvoie le numéro de ligne où est situé le
curseur. Elle n'a besoin d'aucun argument.

Exemple:
    GOTO(1,LINE+1)  ' déplacement au début de la ligne suivante
```

1.121 Fonction ARexx LOAD

```
                  LOAD(nom_fichier)
Charge le fichier texte spécifié dans la fenêtre courante.
La fenêtre perd son contenu, même s'il avait été modifié.
Utilisez la commande
              MODIF
               pour savoir si le texte a été
modifié auparavant, sans avoir été sauvé.
La fenêtre prend le nom du fichier qui a été chargé.
Renvoie 0 en cas d'erreur, sinon 1.

Voir aussi:
              OPEN
              ,
              SAVE
```

1.122 Fonction ARexx LOADREF

```
                  LOADREF(référence)
Demande l'ouverture d'une nouvelle fenêtre, dans laquelle
est chargé le texte spécifique à une
              référence
              .
Renvoie 1 si réussi, 0 sinon.
```

1.123 Fonction ARexx LOCK

```
                  LOCK(x)
Verrouille la fenêtre d'indice spécifié (voir
              SELTEXT
              ). Si x vaut -1
toutes les fenêtres sont verrouillées. C'est à dire que pendant toute
l'exécution du script l'utilisateur ne peut plus avoir accès au
texte: les menus ne sont plus accessibles et les opérations faites
à l'aide du clavier ou de la souris ne sont plus prises en compte.
Si un verrouillage est effectué sur une fenêtre déjà bloquée, un
compteur est incrémenté, il faut ensuite que la fenêtre soit
débloquée autant de fois qu'elle a été bloquée avant qu'elle ne
soit effectivement déverrouillée. Certaines fonctions, comme
le choix d'un fichier, la sauvegarde, le chargement, l'impression,
entraînent un verrouillage temporaire de toutes les fenêtres.
Vous avez généralement intérêt à verrouiller toutes les fenêtres,
sauf condition très particulière (LOCK(-1)). Pour verrouiller
uniquement la fenêtre active, utilisez l'écriture suivante:
    LOCK(SELSHEET(-1))

Valeur renvoyée: code supérieur ou égal à zéro si opération réussie,
    -1 si verrouillage impossible (manque de mémoire ?).

Nota: les fenêtres sont automatiquement débloquées lorsque l'exécution
d'un script se termine. S'il se produit un blocage vous pouvez
```

également mettre fin à cette situation en cliquant deux fois sur
l'icône du programme située sur l'écran du Workbench.
Cette fonction ne peut être appelée depuis une macro clavier (il
y aurait rique de blocage total des entrées/sorties).

Voir aussi:
UNLOCK
.

1.124 Fonction ARexx LOWER

LOWER(chaîne)
Renvoie la chaîne de caractères convertie en minuscules.
La chaîne passée en argument n'est pas modifiée.

Voir aussi:
UPPER

1.125 Fonction ARexx MACRO

MACRO(x)
Exécute la macro associée à l'une des dix touches de fonctions F1 (x=1)
à F10 (x=10). Cette fonction est surtout pratique pour exécuter des
appels de macros entre elles mêmes. Je l'utilise en liaison avec le
menu
Macros/Mode~direct
.
Ainsi la touche F4 peut par exemple appeler la macro associée à la
touche F5 (MACRO(5)).
Valeur renvoyée: résultat du calcul de la macro.
À noter que le type du résultat de la macro peut être quelconque.

1.126 Fonction ARexx MARK

MARK(début/fin)
Permet de sélectionner une zone de texte. Cette fonction doit
être utilisée de la façon suivante:
- placez le curseur au début de la zone à sélectionner
- utilisez la commande MARK(0)
- placez le curseur à la fin de la zone
- utilisez la commande MARK(1)

Les déplacements du curseur seront faits à l'aide de la
fonction GOTO.
Aucune valeur particulière n'est renvoyée par cette fonction.

1.127 Fonction ARexx MENU

```
                  MENU(intitulé menu)
Appelle le menu dont le titre (ou le début du titre seul) est spécifié.
Certains menus sont cependant inopérants pour cette fonction
comme
              Macros
               ou
              Préférences/Fichier
                Il faut aussi savoir que les titres des menus comportant des ↩
                   espaces,
comme "Effacer~fichier" comportent des "espaces solides", obtenus en
frappant ALT-espace au clavier. Ceci a été rendu nécessaire par
AmigaGuide qui ne retrouve pas les noeuds (nodes) dont le nom comporte
des espaces. Il faut donc bien taper le nom du menu avec ces "espaces
solides".
    Ex: MENU ("Quit")   Entraîne la fermeture de tous les tableaux
  MENU ("Copier") Recopie du texte sélectionné dans le presse-papiers

Valeur renvoyée: 1 si le menu a été trouvé, 0 sinon.
```

1.128 Fonction ARexx MESSAGE

```
  MESSAGE(chaîne)
Affiche une requête contenant le message spécifié. Ce texte peut
contenir de une à treize lignes séparées par des sauts de lignes.
    Ex: MESSAGE("Ceci est un"+CHR(10)+"message.")

Valeur renvoyée: 0.
```

1.129 Fonction ARexx MESURE

```
                  MESURE(dimension fenêtre)
Renvoie une des dimensions de la fenêtre courante. La valeur qui
est renvoyée dépend de celle passée en argument:
    égal 0: coordonnée gauche de l'emplacement de la fenêtre
    égal 1: coordonnée haute de l'emplacement de la fenêtre
    égal 2: largeur de la fenêtre (en pixels)
    égal 3: hauteur de la fenêtre (en pixels)
    égal 4: largeur maximale pouvant être prise par la fenêtre
    égal 5: hauteur maximale pouvant être prise par la fenêtre
    égal 6: largeur de l'écran
    égal 7: hauteur de l'écran
Si la fenêtre est cachée la valeur renvoyée est toujours nulle,
sauf pour les dimensions de l'écran.
Si la fenêtre est réduite la valeur déterminant sa dimension
est renvoyée sous forme négative (soit pour les valeurs de
l'argument égales de 0 à 3 compris).

Voir aussi:
              WINDOW
```

1.130 Fonction ARexx MODEDIT

```
  MODEDIT(insertion/surimpression)
Permet de déterminer et/ou de lire le mode d'écriture du
texte dans la fenêtre. Si l'argument est différent de zéro (1)
le mode est le mode insertion, si l'argument est nul c'est le
mode surimpression qui est activé (les caractères écrits
effacent ceux qui s'y trouvaient précédemment).
Renvoie la valeur du mode en cours AVANT l'exécution de
la fonction (permet de réinitialiser le mode à son
ancienne valeur).

Exemple de script:
    'MODEDIT(1)'        passage en mode insertion
    mode=result   lecture de l'ancien mode
    ...     traitement (utilisation WRITE...)
    'MODEDIT('mode')'   retrouver le mode valide au début
```

1.131 Fonction ARexx MODIF

```
  MODIF(fenêtre)
Permet de savoir si une fenêtre contient un texte ayant
été modifié, sans avoir été sauvé. L'argument peut être
positif ou nul, comme renvoyé par la fonction SELTEXT:
c'est l'information concernant cette fenêtre qui est
alors renvoyée. Si l'argument est négatif, c'est pour
la fenêtre courante que l'information est renvoyée.

La valeur renvoyée est nulle si le texte n'a pas été
modifié depuis la dernière sauvegarde, sinon elle est
normalement égale à 1.

Exemple:
    'MODIF(-1)'
    if result~=0 then 'MENU("Sauver")'

    Ce petit script peut être utilisé pour une sauvegarde
    automatique.
```

1.132 Fonction ARexx NBLINES

```
  NBLINES
Cette fonction renvoie le nombre de lignes de texte de
la fenêtre courante. Elle n'a besoin d'aucun argument.

Exemple:
    FOR(L=1:NC=0, L<=NBLINES, NC=NC+LEN(READLINE(L)),L=L+1)
    La ligne ci-dessus permet de compter le nombre total
    de caractères compris dans le texte (variable NC).
```

1.133 Fonction ARexx NBTEXT

```
  NBTEXT(x)
Renvoie le nombre de textes présents en mémoire. Ce
comptage s'effectue sur tout ou partie des fenêtres,
selon la valeur de l'argument:
 égal 0:  on ne compte que les textes cachés
 égal 1:  on ne compte que les textes ouverts (non cachés ou réduits)
 égal -1: on compte tous les textes

Exemple:
    NBTEXT(-1)-NBTEXT(0)    'renvoie le nombre de fenêtres réduites
```

1.134 Fonction ARexx NEW

```
  NEW(titre)
Provoque l'ouverture d'une nouvelle fenêtre, le nom de
cette fenêtre est celui passé en argument. Si l'argument
est une chaîne nulle, la fenêtre prend le nom "Innomé".
Valeur renvoyée: 1 si réussi, 0 sinon.
```

1.135 Fonction ARexx OPEN

```
                  OPEN(nom_fichier)
Cette fonction permet de charger un fichier dans la fenêtre
courante, mais à la différence de la fonction
              LOAD
              , elle
permet également le chargement de plusieurs fichiers. Il
suffit pour cela de spécifier des caractères génériques
(#?[]) dans le nom du fichier, comme prévu sous DOS.
Tous les textes correspondant à ce format seront chargés.
Une nouvelle fenêtre est ouverte pour chacun des textes
trouvés, correspondant à la demande.
La fonction renvoie le nombre de textes ayant pu être chargés.

Exemples:
    OPEN("#?.c")    demande de charger tous les fichiers
        ayant l'extension .c
    OPEN("#?.[chs]")    tous les fichiers possédant les
        extensions .c, .h ou .s sont chargés.
```

1.136 Fonction ARexx PASTE

```
                  PASTE(clip)
Collage du contenu de l'unité du presse-papiers sélectionnée.
Le texte est placé à l'emplacement du curseur. Le numéro de clip
peut varier de 0 à 255. L'unité de clip utilisée pour les
opérations de couper/coller n'est pas modifiée.
```

```
Aucune valeur particulière n'est renvoyée.

Voir aussi:
              COPY
```

1.137 Fonction ARexx PICKCOL

```
                  PICKCOL(message)
Affiche une requête contenant le message spécifié, puis attend
un clic de l'utilisateur dans la fenêtre. Renvoie le numéro de
la colonne où a eu lieu le clic. L'utilisateur peut faire défiler
le texte à l'aide des ascenseurs et des boutons associés.
L'utilisateur peut annuler l'opération en appuyant sur le
bouton droit de la souris ou sur une touche du clavier, la
valeur renvoyée est alors 0.

Voir aussi:
              PICKLINE
```

1.138 Fonction ARexx PICKLINE

```
                  PICKLINE(message)
Affiche une requête contenant le message spécifié, puis attend
un clic de l'utilisateur dans la fenêtre. Renvoie le numéro de
la ligne où a eu lieu le clic. L'utilisateur peut faire défiler
le texte à l'aide des ascenseurs et des boutons associés.
L'utilisateur peut annuler l'opération en appuyant sur le
bouton droit de la souris ou sur une touche du clavier , la
valeur renvoyée est alors 0.

Voir aussi:
              PICKCOL
```

1.139 Fonction ARexx POS

```
  POS(sous_chaîne,chaîne)
Renvoie le rang de la sous-chaîne dans la chaîne, si elle s'y
trouve, sinon renvoie 0.

Exemples:
    POS("s","essai")    renvoie 2
    POS("sa","essai")   renvoie 3
    POS("y","essai")    renvoie 0
```

1.140 Fonction ARexx LOADPREF

```
                  LOADPREF(nom_fichier)
Charge les préférences contenues dans le fichier spécifié.
Ce fichier doit naturellement être au format convenable,
c'est à dire qu'il a dû être créé lors d'une
              sauvegarde
                des préférences.
Renvoie 0 si une erreur se produit, sinon autre valeur.
```

1.141 Fonction ARexx PRINT

```
                  PRINT(ligne début, ligne fin)

              Imprime
               la zone de texte spécifiée.
Renvoie 1 si tout se passe bien, 0 dans le cas contraire.
```

1.142 Fonction ARexx READCHAR

```
  READCHAR(nombre caractères)
Renvoie les caractères courants lus à partir de la
position du curseur. Seuls les caractères de la ligne
courante sont lus: si la ligne se termine avant que
tous les caractères demandés soient lus, la chaîne
renvoyée sera abrégée.

Exemples:
    READCHAR(1) lecture du caractère présent sous le curseur
    READCHAR(2) lecture de deux caractères
```

1.143 Fonction ARexx READLINE

```
  READLINE(ligne)
Renvoie le contenu de la ligne spécifiée.
À noter que le numéro de la première ligne est le 1. La ligne
courante, où est située le curseur, est désignée par la variable
LINE, le nombre total de lignes du texte courant est donné par la
variable NBLINES.

Exemples:
    READLINE(LINE)  lecture de la ligne courante
    READLINE(LINE+1) lecture de la ligne suivante
    READLINE(1)     lecture de la première ligne

Attention: si le curseur se situe après le dernier caractère de
la ligne, les espaces le séparant de la fin de la ligne seront
comptés.
```

1.144 Fonction ARexx REPLACE

```
                  REPLACE(occurences)
Lance la recherche du texte (spécifié par
              SETFIND
               ou par
le menu Rechercher).
Si l'argument occurences est positif, la recherche s'effectue
après l'emplacement du curseur, alors que si cet argument est
négatif, la recherche s'effectue vers le début du texte.
Dans les deux cas la recherche s'arrête quand le nombre
d'occurences spécifié a été atteint. À chaque fois que le
texte cherché a été trouvé, il est remplacé par le texte
spécifié par la fonction
              SETREP
               (ou par celui donné
dans la boîte
              Remplacer
               du menu Recherche).

Pour effectuer un remplacement sur l'ensemble du texte, il suffit par
exemple d'exécuter les lignes qui suivent:
    GOTO(1,1)       ' placement du curseur au début du texte
    SETFIND("Jules")
    SETREP("Léon")
    REPLACE(100000) ' devrait suffire pour remplacer Jules par Léon
          dans tout le texte...

La valeur renvoyée est égale au nombre de fois où la recherche
a abouti.

Voir aussi:
              SETREP
              ,
              SETFIND
              ,
              FIND
              ,
              TESTCASE
              , menu
              Recherche
```

1.145 Fonction ARexx REQFILE

```
  REQFILE(titre)
Ouvre la boîte de requête de fichier, avec le titre spécifié.
Renvoie le nom du fichier sélectionné ou une chaîne nulle
si l'utilisateur clique sur le bouton Annule.
```

1.146 Fonction ARexx REQTEXT

```
                  REQTEXT(titre)
Cette fonction permet de choisir, à l'aide d'une boîte de requête,
un texte parmi ceux qui sont chargés en mémoire. Le titre est
affiché en première ligne, il ne doit pas comprendre de saut de
ligne. Le numéro de texte choisi (de 1 à x, selon le nombre de
textes présents en mémoire) est renvoyé. Si l'utilisateur
appuie sur le bouton droit de la souris, ou bien si un problème
survient une valeur inférieure ou égale à zéro est renvoyée.
Le fonctionnement est analogue à celui obtenu après un double
clic sur le
              bouton droit
               de la souris, cependant avec cette
fonction vous pouvez choisir le titre affiché en en-tête, et
le numéro du bouton choisi est renvoyé.
```

1.147 Fonction ARexx REQUEST

```
  REQUEST(titre)
Affiche une requête avec le texte spécifié (jusqu'à treize
lignes de texte, séparés par des sauts de ligne), et deux
boutons OUI et NON. Si l'utilisateur clique sur le bouton OUI
ou appuie sur ENTRÉE, cette fonction renvoie 1.
Si l'utilisateur clique sur le bouton NON ou appuie sur la
touche ESC, la valeur renvoyée est 0.
En cas de problème (manque mémoire par exemple), la valeur
renvoyée est négative.
```

1.148 Fonction ARexx RESET

```
                  RESET(variable,...)
    Cette fonction est identique à la fonction
              INIT
              ,
cependant le type n'est pas réinitialisé.
Si la
              variable
               est du type numérique elle prend la valeur 0, si
c'est une chaîne de caractères elle devient une chaîne vide ("").
    Le nombre d'arguments est quelconque.
```

1.149 Fonction ARexx SAVE

```
                  SAVE(nom_fichier)
Sauve le texte sous le nom spécifié. Si le fichier existait déjà
aucun avertissement n'a lieu. Renvoie 1 si aucun problème ne
survient, sinon 0. Un fichier .info comportant une icône est
créée si le menu
              Préférences/Sauver~icône
               est marqué, un fichier
```

comportant l'extension .bis est créé si le menu

Préférences/Copie texte sauvé
est marqué.

1.150 Fonction ARexx SAVBLOCK

SAVBLOCK(nom_fichier)
Sauve le texte marqué dans le fichier spécifié.
Renvoie 1 si tout se passe bien, 0 en cas de problème, -1 si
aucun texte n'est marqué.

1.151 Fonction ARexx SAVECOPY

SAVECOPY(1/0/-1)
Cette fonction permet de fixer, comme le menu
Préférences/Copie~texte~sauvé
,
si une copie du texte sera créée lors d'une opération de sauvegarde
du texte.
Cette fonction renvoie 1 si le menu était marqué AVANT l'exécution
de la fonction, ou 0 s'il ne l'était pas.
Pour lire l'état du menu sans le modifier, utilisez la commande
SAVECOPY(-1).

1.152 Fonction ARexx SAVEICON

SAVEICON(1/0/-1)
Cette fonction permet de fixer, comme le menu
Préférences/Sauver~icône
,
si une icône sera créée lors d'une opération de sauvegarde
du texte.
Cette fonction renvoie 1 si le menu était marqué AVANT l'exécution
de la fonction, ou 0 s'il ne l'était pas.
Pour lire l'état du menu sans le modifier, utilisez la commande
SAVEICON(-1).

1.153 Fonction ARexx SAVEPREF

SAVEPREF("nom fichier")
Cette fonction permet de sauvegarder les préférences courantes sous
le nom spécifié. Elle renvoie 1 si tout se passe bien, 0 dans le cas
contraire

1.154 Fonction ARexx SAVETABS

```
                  SAVETABS(1/0/-1)
Cette fonction permet de fixer, comme le menu
              Préférences/Sauver tabulation
              ,
si les tabulations seront sauvées comme telles ou remplacées par des
espaces, lors d'une opération de sauvegarde du texte.
Ainsi SAVETABS(1) valide la sauvegarde avec tabulations alors que
SAVETABS(0) l'annule.
Cette fonction renvoie 1 si le menu était marqué AVANT l'exécution
de la fonction, ou 0 s'il ne l'était pas.
Pour lire l'état du menu sans le modifier, utilisez la commande
SAVETABS(-1).

Exemple de script:
    'SAVETABS(0)'       sauver avec des espaces (code ASCII pur)
    mode=result   lecture de l'ancien mode
    ...     traitement (utilisation SAVE...)
    'SAVETABS('mode')'  retrouver le mode valide au début
```

1.155 Fonction ARexx SEARCH

```
  SEARCH(chaîne)
Cette fonction permet de rechercher une chaîne de caractères
comprenant des caractères génériques dans le texte.
La recherche s'effectue toujours à partir de la position
courante du curseur, vers la fin du texte. Si le texte
cherché ne termine pas la ligne vous devez impérativement
terminer la chaîne par les caractères #? (ou *), la
comparaison s'effectuant toujours sur la ligne entière.
Par ailleurs la comparaison différencie toujours les
majuscules et les minuscules.
La fonction renvoie 1 si la chaîne a été trouvée, le curseur
se positionnant où a été trouvée cette chaîne. En cas d'échec
la fonction renvoie 0.

Exemple:
    SEARCH("text#?")    cherche un mot commençant par "text"
```

1.156 Fonction ARexx SECURITY

```
                  SECURITY(nombre boucles)
Cette fonction détermine le nombre maximal de boucles pouvant
être effectuées par une des fonctions
              FOR
               ou
              WHILE
              . Ceci
permet de sortir des boucles sans fin assez simplement.
La valeur par défaut est égale à 500. Vous pouvez entrer
une valeur allant de 1 jusqu'à 2^31-1 (2147483647). La
```

```
fonction renvoie la valeur en cours avant son exécution.

Vous pouvez donner à cette fonction un argument de grande
valeur sans craindre de blocage, il est effet maintenant
possible d'interrompre une boucle en appuyant simultanément
sur les touches CTRL, ALT et ESC.

Si la valeur passée en argument est nulle seule la valeur
actuelle est renvoyée, sans modification.
```

1.157 Fonction ARexx SELECT

```
  SELECT(texte)
Cette fonction permet d'ouvrir une boîte de requête comportant
un nombre variable de boutons, comportant chacun un texte
choisi par l'utilsateur. Si l'utilisateur clique sur l'un
de ces boutons, la fonction renvoie le rang du bouton (en
commençant par la valeur 1 pour le premier, 2 pour le
second, etc...). Il peut y avoir jusqu'à 13 boutons.
Le format du texte passé en argument doit être le suivant:
    - une première ligne, affichée en titre,
    - une seconde ligne, correspondant au texte du premier bouton
    - une troisième ligne, correspondant au texte du second bouton,
    - et ainsi de suite, autant de lignes que de boutons...
Chacune des lignes est séparée des suivantes par un saut de
ligne (CHR(10)).
Si la valeur renvoyée est négative ou nulle, c'est que
l'utilisateur a appuyé sur le bouton droit ou sur la touche
Esc, ou bien que la requête n'a pu être ouverte.

Exemple:
    SELECT("Nombre de lignes ?"+CHR(10)+"10 lignes"+CHR(10)+"20 lignes"+CHR(10) ↩
       +"À déterminer")
```

1.158 Fonction ARexx SELFILE

```
                  SELFILE("nom_fichier")
Cette fonction permet de choisir la fenêtre active, en
spécifiant son nom. Le nom spécifié peut être le nom
complet, ou bien le nom du fichier, sans le chemin.
Si plusieurs fenêtres possèdent le même nom, la première
fenêtre trouvée est sélectionnée.
Cependant la fenêtre sélectionnée ne passe pas au
premier plan, utilisez pour cela la fonction
              TOFRONT
              .
De même si la fenêtre est réduite ou bien cachée, elle le
restera. Ceci est utile pour lire une ligne ou quelques
caractères d'une fenêtre sans la "réveiller".

La valeur renvoyée est positive ou nulle si la fenêtre a
été trouvée (elle correspond à la valeur qu'aurait renvoyé
```

l'instruction
SELTEXT
(-1)), sinon elle est négative.

Exemples:
SELFILE("RAM:texte")
SELFILE("texte")
Ces deux exemples permettent la sélection de la même fenêtre (RAM:texte).

1.159 Fonction ARexx SELTEXT

SELTEXT(fenêtre)
Cette fonction permet de choisir la fenêtre active ou de connaître l'indice de la fenêtre courante.
Si l'argument est positif ou nul, la fenêtre d'indice spécifié est sélectionnée et devient la fenêtre active.
Cependant la fenêtre sélectionnée ne passe pas au premier plan, utilisez pour cela la fonction
TOFRONT
.
De même si la fenêtre est réduite ou bien cachée, elle le restera. Ceci est utile pour lire une ligne ou quelques caractères d'une fenêtre sans la "réveiller".

La fonction renvoie toujours l'indice de la fenêtre active avant son exécution. Si l'indice passé en argument est négatif, seule cette valeur est renvoyée.

1.160 Fonction ARexx SETFIND

SETFIND(chaîne)
Détermine quelle est la chaîne de caractères à chercher par les menu
Recherche/Avant/Après
ou
Remplacer
, ainsi que par les fonctions
FIND
et
REPLACE
.
Renvoie la chaîne de caractères précédemment recherchée.

1.161 Fonction ARexx SETREP

SETREP(chaîne)
Détermine quelle est la chaîne de caractères de remplacement utilisée par le menu

```
              Recherche/Remplacer
              , ainsi que par la
fonction
              REPLACE
              .
Renvoie la chaîne de caractères qui était utilisée précédemment.
```

1.162 Fonction ARexx SETFONT

```
  SETFONT(fonte,taille)
Fixe la fonte utilisée pour afficher le texte dans la
fenêtre courante. Le nom de la fonte doit être complet,
y compris l'extension ".font".
Renvoie 1 si la fonte a pu être chargée, sinon 0.

Exemple:
    SETFONT("topaz.font",11)
```

1.163 Fonction ARexx SETTABS

```
                  SETTABS(tabulations)
Fixe le nombre d'espaces par tabulation. Cette fonction
renvoie la valeur qui était utilisée AVANT qu'elle ne soit
exécutée. La valeur de l'argument doit être comprise entre
1 et 39 compris pour être prise en compte. Une valeur
inférieure n'entraîne aucune modification mais permet
la lecture de la valeur courante.

Exemple de script:
    'SETTABS(8)'
    tabs=result   lecture de l'ancien mode
    ...     traitement (utilisation
              WRITETAB
              ...)
    'SETTABS('tabs')'   retrouver la valeur valide au début
```

1.164 Fonction ARexx SGN

```
  SGN(nombre)
Renvoie 1 si l'argument est positif, -1 s'il est négatif,
zéro s'il est nul.
```

1.165 Fonction ARexx SORT

```
  SORT(sens, ligne_début, ligne_fin)
Cette fonction permet de trier les lignes spécifiées, de
les classer dans l'ordre alphabétique ou inverse.
```

Si l'argument sens est égal à 1, le tri a lieu dans l'ordre
croissant, s'il est égal à -1, le tri se fait dans l'ordre
décroissant. Les comparaisons s'effectuent en commençant
par la première colonne, même s'il s'agit d'un espace.

À noter que le numéro de la première ligne est le 1. La ligne
courante, où est située le curseur, est désignée par la variable
LINE, le nombre total de lignes du texte courant est donné par la
variable NBLINES.

Cette fonction renvoie le nombre de permutations qu'elle a dû
effectuer pour réaliser l'opération.

1.166 Fonction ARexx STR

STR(nombre)
Renvoie la chaîne de caractères correspondant à un nombre.
La base utilisée pour la conversion est la base 10 (décimal).

1.167 Fonction ARexx STRBIN

STRBIN(nombre)
Renvoie la chaîne de caractères correspondant à un nombre.
La base utilisée pour la conversion est la base 2 (binaire).
La conversion est effectuée systématiquement sur 32 bits.

Exemple:
    STRBIN(30) renvoie la chaîne "00000000000000000000000000011110"

1.168 Fonction ARexx STRHEX

STRHEX(nombre)
Renvoie la chaîne de caractères correspondant à un nombre.
La base utilisée pour la conversion est la base 16 (hexadécimal).

Exemple:
    STRHEX(30) renvoie la chaîne "1E"

1.169 Fonction ARexx SUBS

SUBS(chaîne, position, longueur)
Renvoie un sous-ensemble de la chaîne passée en argument.
Ce sous-ensemble commence au caractère spécifié par le
second argument, et comprend le nombre de caractères donné
par le troisième argument.

Exemple:

```
    SUBS("Texte",3,2)   renvoie xt

Vous pouvez définir les fonctions classiques RIGHT et LEFT
à partir de cette fonction:
    DEF RIGHT(CHAÎNE,N) = SUBS(CHAÎNE,LEN(CHAÎNE)-N+1,N)
    DEF LEFT(CHAÎNE,N) = SUBS(CHAÎNE,1,N)
```

1.170 Fonction ARexx SUPBLOCK

```
  SUPBLOCK(0)
Cette fonction permet d'effacer le bloc de texte marqué.
La valeur renvoyée est 1 si tout se passe normalement,
0 si aucun bloc de texte n'était sélectionné.
Attention, le bloc de texte ne peut être récupéré après
cette opération. L'argument est sans importance, passez
un zéro de préférence.
```

1.171 Fonction ARexx SUPLINES

```
  SUPLINES(nombre lignes)
Suppression du nombre de lignes spécifié, en commençant
à la ligne où est situé le curseur.
Renvoie le nombre de lignes restant dans la fenêtre.
```

1.172 Fonction ARexx TESTCASE

```
  TESTCASE(1/0/-1)
Permet de déterminer si les opérations de recherche
tiendront compte ou non du test minuscules/MAJUSCULES.
Si l'argument est nul, les majuscules et les minuscules
sont différenciées, s'il est égal à 1 les majuscules et
les minuscules sont considérées comme égales, enfin
s'il est négatif le test n'est pas changé, seule la
valeur correspondant au mode actuel est renvoyée.

Exemple de script:
    'TESTCASE(1)'       rend le test indifférencié (MAJ=min)
    mode=result
    ...     recherches/remplacements...
    'TESTCASE('mode')'  on remet le mode tel qu'il était
```

1.173 Fonction ARexx TEXTMARK

```
  TEXTMARK(-1)
Renvoie le texte correspondant au bloc sélectionné. Si
le texte tient sur plusieurs lignes, celles-ci sont
séparées par des sauts de ligne.
```

```
L'argument n'a aucune importance (passez plutôt -1, en
vue d'une future compatibilité).
```

1.174 Fonction ARexx TIME

```
                  TIME(secondes)
Renvoie l'heure courante, si l'argument est non nul, les
secondes sont comprises.

Exemples:
    TIME(0)     renvoie 23:24
    TIME(1)     renvoie 23:24:27

Voir aussi:
              DATE
```

1.175 Fonction ARexx TITLE

```
  TITLE(titre)
Spécifie le titre de la fenêtre courante. Si l'argument
est une chaîne nulle, le titre reprend son état normal,
c'est à dire que le nom du fichier y réapparaît.
Cette fonction permet de signaler une opération un peu
longue en prévenant l'utilisateur, sans pour autant
bloquer la fenêtre ni le programme.

Exemple de script:
    'TITLE("Traitement en cours...")'
    ...     traitement...
    'TITLE("")'         on remet le titre normal
```

1.176 Fonction ARexx TOBACK

```
                  TOBACK(fenêtre)
Envoie la fenêtre d'indice spécifié ou la fenêtre
courante (argument égal à -1) en arrière-plan.

Voir aussi:
              SELTEXT
              ,
              TOFRONT
```

1.177 Fonction ARexx TOFRONT

```
                  TOFRONT(fenêtre)
Envoie la fenêtre d'indice spécifié ou la fenêtre
courante (argument égal à -1) à l'avant-plan.
```

```
Voir aussi:
              SELTEXT
              ,
              TOBACK
```

1.178 Fonction ARexx TRACE

```
  TRACE(marque)
Permet de mémoriser l'emplacement actuel du curseur.
10 marques peuvent être ainsi mémorisées. Elles correspondent
aux marques obtenues à l'aide des touches de fonction
associées à la touche CTRL. L'argument peut ainsi prendre
les valeurs de 1 à 10. S'il est négatif le curseur est
ramené à l'emplacement précédemment marqué.
```

1.179 Fonction ARexx TYPE

```
                  TYPE(caractère)
Cette fonction permet de connaître la nature d'un caractère.
L'argument passé à la fonction doit être le code d'un caractère,
obtenu par la fonction
              ASC
              .
La valeur renvoyée dépend du type du caractère:
    - égale à 1: c'est une voyelle,
    - égale à 2: c'est une lettre de l'alphabet,
    - égale à 4: c'est une majuscule,
    - égale à 8: c'est une minuscule,
    - égale à 16: c'est un nombre décimal,
    - égale à 32: c'est un espace,
    - égale à 64: c'est un signe de ponctuation,
    - égale à 128: c'est un caractère ASCII standard,
    - égale à 256: c'est un caractère pouvant faire partie d'un
  nom de
              variable
               ou de fonction.
Ces valeurs peuvent être combinées entre elles, ainsi TYPE(ASC("e",1))
renvoie la valeur 395, soit 256 + 128 + 8 + 2 + 1.
Pour savoir si le résultat correspond à une valeur particulière, il
suffit de réaliser une ET logique entre le résultat et la valeur
cherchée, ainsi pour savoir si un caractère quelconque est une
voyelle, vous pouvez définir la fonction suivante:
    DEF VOYELLE(C) = IF (TYPE(C) & 1, 1, 0)
Pour savoir s'il s'agit d'une lettre majuscule:
    DEF MAJUSCULE(C) = IF (TYPE(C) & 4, 1, 0)
Et ainsi de suite pour les fonctions suivantes:
    DEF MINUSCULE(C) = IF (TYPE(C) & 8, 1, 0)
    DEF ALPHA(C) = IF (TYPE(C) & 2, 1, 0)
    DEF NOMBRE(C) = IF (TYPE(C) & 16, 1, 0)
    DEF ESPACE(C) = IF (TYPE(C) & 32, 1, 0)
    DEF PONCTUATION(C) = IF (TYPE(C) & 64, 1, 0)...
```

Chacune de ces fonctions renvoie 1 si le caractère est du
type testé, sinon elles renvoient 0.

1.180 Fonction ARexx UNLOCK

UNLOCK(fenêtre)
Annule le verrouillage d'une fenêtre, réalisé par la
fonction
LOCK
.
Si l'argument est positif ou nul, seule la fenêtre
possédant l'indice spécifié est déverrouillée, si
cet argument est négatif (-1), toutes les fenêtres
sont deverrouillées.

Voir aussi:
SELTEXT
.

1.181 Fonction ARexx UNMARK

UNMARK(-1)
Annule le marquage de la zone de texte sélectionnée.
L'argument n'est pas pris en compte, passez un -1 de
préférence (compatibilité éventuelle avec les version
ultérieures).
Aucune valeur particulière n'est renvoyée.

1.182 Fonction ARexx UPPER

UPPER(chaîne)
Renvoie la chaîne de caractères convertie en majuscules.
La chaîne passée en argument n'est pas modifiée.

Voir aussi:
LOWER

1.183 Fonction ARexx VAL

VAL(chaîne)
Renvoie la valeur numérique correspondant à la chaîne de
caractères passée en argument. Seuls les caractères
correspondant à des nombres entiers sont pris en compte.
Le nombre doit figurer sous forme décimale, sauf s'il est
précédé du préfixe $, qui signale qu'il s'agit d'un
nombre sous forme hexadécimale.

```
Exemples:
    VAL("14")       renvoie 14
    VAL("14.3")     renvoie 14
    VAL("$5F")      renvoie 95 (le $ spécifie une chaîne hexadécimale)
```

1.184 Fonction ARexx VERSION

```
  VERSION(type)
Cette fonction renvoie le numéro de version du programme ou
un message de copyright, selon la valeur de l'argument. Le
test du numéro de version peut être intéressant pour interdire
l'exécution d'un script à un programme trop ancien, qui risque
de mal exécuter certaines instructions qui ont évolué dans les
versions plus récentes.

Exemples (résultats différents selon la version du programme):
    VERSION(0)      renvoie 3.00
    VERSION(1)      renvoie 68020 (ou 68000)
    VERSION(2)      renvoie © R.FLORAC 27 décembre 1996

Exemple de script:
    'VERSION(0)'
    if result < 3.00 then do
  'MESSAGE("Cette version du programme"+CHR(10)+"ne convient pas.")'
  exit
    end
    ...
```

1.185 Fonction ARexx WHILE

```
                  WHILE(fin,action1,...)
Cette fonction est identique à la fonction
              FOR
              , il
lui manque juste le premier argument. L'initialisation des

              variables
               ou des cellules testées aura donc due être effectuée
auparavant.

Voir aussi:
              SECURITY
```

1.186 Fonction ARexx WINDOW

```
                  WINDOW(x, y, largeur, hauteur)
Cette fonction permet de redimensionner la fenêtre courante, ou
de la déplacer.
Les deux premiers arguments correspondent aux coordonnées du coin
supérieur gauche de la fenêtre dans l'écran. Les deux arguments
```

```
suivants déterminent ses dimensions.
Attention: cette fonction ne fonctionne que pour les fenêtres
"normales".Pour les fenêtres réduites, seules les coordonnées
x et y peuvent être modifiées (pour déplacer les fenêtres).
Les fenêtres cachées ne peuvent naturellement pas être modifiées,
mais il suffit de spécifier la commande WINDOW(-1,-1,-1,-1) pour
permettre leur réouverture.
Si vous ne voulez pas modifier un ou plusieurs de ces paramètres
et que vous ne connaissiez pas leur valeur, donnez-leur une valeur
négative (-1). Si les arguments ont des valeurs trop grandes ou
aberrantes, le programme essaiera d'y remédier au mieux.
La valeur renvoyée est égale au nombre de valeurs ayant été prises
en compte.

Exemples:
    WINDOW(0,0,-1,-1)   déplace la fenêtre en haut à gauche de l'écran
      sans modifier ses dimensions.
    WINDOW(-1,-1,200,50) change les dimensions de la fenêtre

Voir aussi:
              MESURE
              ,
              type d'outil WINDOW
```

1.187 Fonction ARexx WLEFT

```
                  WLEFT(nombre mots)
Déplace le curseur du nombre de mots spécifié vers la gauche.
Si le curseur atteint le début de la ligne, le déplacement se
poursuit à la fin de la ligne qui précède.
Renvoie le numéro de ligne atteint.

Voir aussi
              WRIGHT
```

1.188 Fonction ARexx WORD

```
  WORD(0)
Renvoie le mot courant. L'argument est sans importance.
Si le caractère présent sous le curseur n'est pas un
caractère appartenant à un mot, une chaîne nulle est
renvoyée. Le mot commence à l'emplacement du curseur,
les caractères qui précèdent sont ignorés.
```

1.189 Fonction ARexx WRIGHT

```
                  WRIGHT(nombre mots)
Déplace le curseur du nombre de mots spécifié vers la droite.
Si le curseur atteint la fin de la ligne, le déplacement se
poursuit au début de la ligne qui suit.
```

Renvoie le numéro de ligne atteint.

Voir aussi
WLEFT

1.190 Fonction ARexx WRITE

WRITE(chaîne)
Écriture d'une chaîne de caractères à l'emplacement du curseur. Cette fonction utilise le
mode courant
(insertion
ou surimpression). Elle renvoie le nombre de caractères ayant été effectivement écrits.
La chaîne de caractères passée en arguments peut contenir des sauts de ligne (
CHR
(10)), ils seront pris en compte.

1.191 Fonction ARexx WRITETAB

WRITETAB(nombre tabulations)
Permet d'aligner le curseur sur les tabulations qui suivent. Si le nombre de tabulations spécifié est positif, des espaces sont écrits, alors que s'il est négatif, le curseur est déplacé, sans écrire. Les tabulations sont définies par le menu
Préférences/Tabulations~=
Renvoie le numéro de colonne atteint.

1.192 Commandes clavier

HELP Lance le programme AmigaGuide, celui-ci charge le fichier d'aide Amitex.guide. Ce fichier guide doit se trouver dans le répertoire où est situé le programme AmiTex.
Si AmigaGuide n'est pas disponible, le script AideMacros.amitex sera lancé, s'il existe.

Commandes d'édition

RETURN Insertion d'une nouvelle ligne sous la ligne actuelle. Le curseur se place à la hauteur du début de la ligne précédente.
ENTER Coupure de la ligne, à l'endroit où est situé le curseur
SHIFT-RETURN, SHIFT-ENTER
Coupure de la ligne, alignement de la seconde ligne sous le premier caractère de la ligne précédente.
DEL Effacement du caractère situé sous le curseur
BACKSPACE Effacement du caractère situé avant le curseur

CTRL DEL Effacement de la ligne
ALT DEL Restauration de la ligne détruite (non nulle) par CTRL DEL
SHIFT DEL Effacement de la fin de la ligne
SHIFT BACKSPACE Effacement du début de la ligne

FLÈCHES déplacement curseur

FLÈCHES DE DÉPLACEMENT : GAUCHE / DROITE / HAUT / BAS

ALT FLÈCHE   DROITE/GAUCHE : Déplacement mot par mot
SHIFT FLÈCHE DROITE/GAUCHE : Déplacement sur le premier ou le dernier caractère de la ligne
CTRL FLÈCHE DROITE/GAUCHE : Déplacement sur la première ou la dernière colonne de la ligne.

ALT   FLÈCHE BAS/HAUT : Défilement du texte ligne par ligne
SHIFT FLÈCHE BAS/HAUT : Défilement du texte page écran par page écran
CTRL FLÈCHE BAS/HAUT : Déplacement au début ou à la fin du texte

SHIFT ALT FLÈCHE HAUT : Déplacement du curseur sur la première colonne de la première ligne de la fenêtre.
SHIFT ALT FLÉCHE BAS : Déplacement du curseur sur la première colonne de la dernière ligne de la fenêtre.

TAB   : Déplacement sur la tabulation qui suit
CTRL TAB : Insertion d'espaces jusqu'à la tabulation suivante
SHIFT TAB: Passage à la tabulation précédente
ALT TAB : Effacement des caractères jusqu'à la tabulation précédente

CTRL HELP: Déplacement du curseur au début du bloc spécifié
SHIFT HELP: Déplacement du curseur à la fin du bloc spécifié
ALT HELP : affichage succinct du rôle des touches ALT avec les touches de fonctions.

La combinaison CTRL-ESC permet de retrouver la dernière ligne ayant été modifiée, dans la dernière fenêtre ayant subi cette modification. Cette fenêtre est ramenée au premier plan si elle est cachée.
La combinaison SHIFT-ESC permet la même chose mais DANS la fenêtre courante. Si cette dernière n'a pas été modifiée, le curseur va à la première ligne.

Utilisation des touches de FONCTION

F1   : réduction de la fenêtre à sa taille minimale
F2   : agrandissement de la fenêtre à sa taille maximale
F3   : ouverture d'une nouvelle fenêtre, choix du fichier à charger
F4   : ouverture d'une nouvelle fenêtre
F5   : recherche occurence précédente (idem AMIGA-B)
F6   : recherche occurence suivante (idem AMIGA-N)
F7   : effacement du mot à gauche (ou des espaces)
F8   : effacement du mot à droite du curseur (ou des espaces)
F9   : déplacement de l'écran en arrière
F10   : déplacement de la fenêtre en arrière

SHIFT F1/F10 : marquage de l'emplacement actuel du curseur

```
CTRL  F1/F10 : déplacement du curseur à l'emplacement précédemment marqué

ALT F1/F10   : programmation d'une macro-commande, puis répétition.
SHIFT ALT F1/F10: reprogrammation des macro-commandes.
```

1.193 bgui.library

```
L'auteur de la bibliothèque bgui.library est Jan van den Baard.

    BGUI release 1.1
    (C) Copyright 1993-1994  Jaba Development
    (C) Copyright 1993-1994 Jan van den Baard
     Écrit avec le compilateur DICE v3.0 par

  Poste: Jan van den Baard
         Bakkerstraat  176
         3082 HE Rotterdam
        Holland

     Modem: 2:286/407.53  (Jan van.den.Baard)
     EMail: jaba@grafix.wlink.nl
```

1.194 Aide en ligne

```
                Une aide en ligne peut être obtenue à tout moment, à l'aide du
fichier AmiTex.guide. Ce fichier doit se situer dans le même
répertoire que le programme ou bien dans à un emplacement et
un nom spécifiés dans le type d'outil
            HELPFILE
             de l'icône du
programme (cette information n'est prise en compte que lors
du lancement du programme par le Workbench et non depuis un
Shell).

Pour lancer l'aide vous pouvez appuyer sur la touche HELP,
à tout moment ou bien lors de la sélection d'un menu, ce qui
vous permet d'avoir directement l'aide concernant ce menu.
Vous pouvez aussi utiliser le menu
            Projet/Aide spécifique
                et donner le nom d'un noeud (node) du fichier d'aide. Vous
pouvez ainsi trouver une aide très rapidement pour n'importe
quelle fonction, en donnant son nom.

À noter que vous pouvez également provoquer l'apparition de l'aide
sur une fonction donnée, si une erreur survient, provoquée par
cette fonction. Appuyez simplement sur la touche HELP alors que
la requête signalant l'erreur est encore affichée.

Vous pouvez également rechercher l'aide d'une fonction ou d'un
point quelconque en sélectionnant ce terme dans le texte, puis
en appuyant sur la touche HELP. Cependant cette recherche
n'aboutira que s'il existe un node de ce nom dans le fichier
```

d'aide.

Si, pour une raison ou pour une autre, l'aide ne peut être
lancée, vous pouvez bénéficier d'une aide moins performante
mais substantielle en lançant l'exécution du script AideMacros.

1.195 Références

Les références sont utilisées principalement par les programmeurs
pour trouver rapidement une aide sur une fonction ou une structure
manipulée par le système. Il s'agit d'un embryon de système hypertexte.

Ces références sont recherchées dans le fichier "s:références_edit".
Ce fichier possède bien sûr un format spécifique, il s'agit cependant d'un
fichier au format ASCII pouvant être édité par le programme AmiTex lui-même.
Chacune des lignes de ce fichier est relative à une référence.
A partir de la PREMIÈRE colonne figure la référence proprement dite,
celle qui peut être trouvée en sélectionnant le menu "Référence" (AMIGA-T).
La casse des lettres (majuscules ou minuscules) est importante, il
faut en tenir compte.
Un espace (et un seul) doit délimiter ce champ avec le suivant,
qui détermine le nombre de lignes devant être lues dans le fichier
documentation (si c'est un nombre) ou bien quelle est la chaîne de
caractères qui déterminera l'arrêt de la lecture dans ce fichier.
Cette chaîne doit être encadrée par des parenthèses si elle comprend
des espaces.
Le champ qui suit (toujours séparé par UN espace) détermine le
fichier documentation à lire (avec le chemin complet).
Enfin le reste de la ligne détermine le dernier champ, qui définit
lui-même la chaîne de caractères à chercher dans le fichier
documentation pour débuter sa lecture (cette chaîne devra figurer dans
la première colonne du fichier documentation). Si cette chaîne
n'existe pas le fichier sera chargé depuis le début (première ligne).

Exemples :

Supposons que le fichier références_edit contienne la ligne qui
suit:

```
RectFill 36 DH1:docs/graphics RectF
```

Le mot clé est RectFill : c'est à dire qu'en positionnant le
curseur sur le début de ce mot dans un texte, puis en sélectionnant
le menu Référence (ou AMIGA-T) le contenu du fichier DH1:docs/graphics
sera chargé dans une fenêtre, et ceci en commençant à la ligne débutant
par les caractères RectF (le fichier peut ainsi contenir plusieurs
références accessibles différemment). 36 lignes de texte seront lues.

Vous pouvez vous reporter au fichier joint (dans le répertoire s:)
pour voir quelques exemples.
Vous pouvez également utiliser le script ARexx AjoutRef.Amitex pour
ajouter automatiquement des références à votre fichier de références.
Vous devrez cependant adapter celui-ci à votre configuration.

1.196 BUG(s) ? (mais oui ! sûrement... (malheureusement !))

Ce programme m'a demandé beaucoup de travail aussi je vous demanderais de bien vouloir être indulgent quant aux erreurs toujours possibles survenant lors de son utilisation. Je vous saurais gré de bien vouloir
m'avertir
si vous constatez une ou des anomalies.

1.197 AMÉLIORATIONS POSSIBLES

Les améliorations suivantes seront faites si le besoin s'en fait sentir et si j'en ai le temps (et aussi si j'en ai le courage !) :
- choix de l'icône créée par le programme lors d'une sauvegarde,
- ajouter la possibilité de "replier" des parties de texte,
- ajouter la possibilité de visualiser deux parties d'un même texte,
- sauvegarde des préférences dans le répertoire ENVARC:,
- modification du programme pour en faire un traitement de textes à part entière : gestion des styles (italique, souligné...) et formatage du texte (marges, paragraphes, etc).

1.198 L'auteur

Pour me contacter:

FLORAC Roland
6 Rue des Chardonnerets
Chez Corbin
17610 Chaniers
Tél: 05 46 93 95 71

1.199 Index

LINE

Liste alphabétique des fonctions ARexx

Liste thématique des fonctions ARexx

LOAD

LOADPREF

LOADREF

LOCK

LOWER
M

MACRO

Macro-commande

MARK

MENU

Menu Macros

Menu Projet

Menu Préférences

Menu Recherche

Menu Édition

MESSAGE

MESURE

MODEDIT

MODIF
N

NBLINES

NBTEXT

NEW
O

OPEN
P

PASTE

PICKCOL

PICKLINE

Port ARexx

POS

PRINT
R

READCHAR

READLINE

Recherche du texte sélectionné

REPLACE

REQFILE

REQTEXT

REQUEST

RESET

Références
S

SAVBLOCK

SAVE

SAVECOPY

SAVEICON

SAVEPREF

SAVETABS

Scripts ARexx

SEARCH

SECURITY

SELECT

SELFILE

SELTEXT
SETCLIP

SETFIND

SETFONT

SETREP

SETTABS

SGN

SORT

STARTUP

STR

STRBIN

STRHEX

SUBS

SUPBLOCK

SUPLINES
T

TESTCASE

TEXTMARK

TIME

TITLE

TOBACK

TOFRONT

TRACE

TYPE

Type d'outil WINDOW
U

UNLOCK

UNMARK

UPPER

V

VAL

Valeurs numériques

Variables

VERSION
W

WHILE

WINDOW

WLEFT

WORD

WRIGHT

WRITE

WRITETAB
X

X_ICON
Y

Y_ICON